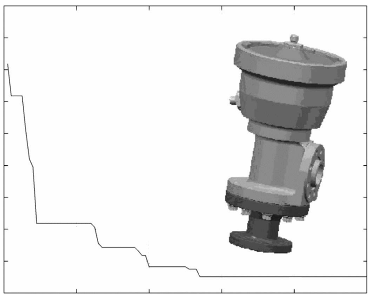
UNIVERSITÉ LIBRE DE BRUXELLES
FACULTÉ DES SCIENCES APPLIQUÉES

Multicriteria Optimization with Expert Rules for Mechanical Design



Rajan Filomeno Coelho

Dissertation originale présentée en vue de l'obtention
du grade de Docteur en Sciences Appliquées

Année académique 2003 – 2004

Multicriteria Optimization with Expert Rules for Mechanical Design

Rajan Filomeno Coelho

Co-promoteurs :

Pr. Philippe Bouillard
Structural and Material
Computational mechanics department (ULB)

Pr. Hugues Bersini
IRIDIA
(ULB)

Thèse de doctorat financée dans le cadre de la convention FIRST Doctorat MINOS (*Moteur d'INférence pour l'Optimisation de Structures*) n° 001/4442 par la Région Wallonne et Samtech s.a.

RÉSUMÉ

Alors que de nombreuses méthodes numériques ont été proposées dans la littérature pour l'optimisation de structures lors des dernières étapes de la conception, peu d'ingénieurs concepteurs utilisent ces outils dès le début d'un projet. Cependant, une légère modification au début du processus peut fournir des améliorations sensibles aux performances globales de la structure.

Pour réaliser l'optimisation d'un design en phase d'avant-projet, le choix s'est porté sur les algorithmes évolutionnaires, notamment en raison de leur capacité à explorer largement l'espace de recherche. Néanmoins, ils ont des difficultés à traiter efficacement des contraintes. De plus, dans les applications industrielles, on rencontre souvent plusieurs objectifs à satisfaire simultanément. Partant de ces constatations, une méthode originale a été proposée : PAMUC (Préférences Appliquées au MUltiobjectif et aux Contraintes). Dans un premier temps, l'utilisateur doit attribuer des poids à chaque objectif. Ensuite, une fonction objectif supplémentaire est créée en agrégeant linéairement les contraintes normalisées. Enfin, une méthode issue de l'aide multicritère à la décision, PROMETHEE II, est utilisée pour classer les individus de la population de l'algorithme évolutionnaire. PAMUC a été validée sur des cas tests multiobjectifs standards, et implantée ensuite dans le logiciel Boss Quattro (Samtech s.a.). La méthode a été utilisée pour l'optimisation paramétrique de vannes à clapet équipant le moteur VINCI d'Ariane 5 et développées par Techspace Aero (Snecma group).

La seconde partie de la thèse a consisté à incorporer un moteur d'inférence au sein du processus d'optimisation afin de prendre en compte des règles d'expert. Au départ, l'information concernant la conception et le dimensionnement (ayant généralement trait à des contraintes technologiques d'usinage, d'assemblage, etc.) doit être collectée auprès d'experts dans le domaine considéré, et traduites sous forme de règles logiques. Ensuite, chaque design potentiel généré par l'algorithme évolutionnaire est testé et éventuellement réparé s'il ne respecte pas les règles d'expert. Cette approche, appliquée à des benchmarks de mécanique et aux vannes à clapet précitées, s'avère efficace pour réduire la taille de l'espace de recherche et guider l'algorithme vers l'optimum global (respectant les contraintes).

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

A PhD thesis is far from being a “long and quiet river” ; actually, it could not possibly be achieved without the help and support of several people.

First of all, I would like to thank warmly Pr. Philippe Bouillard for his help, advice and continuous support throughout the duration of my PhD. His enthousiasm and energy have been very profitable over the past few years. I am also grateful to Pr. Hugues Bersini, for his precious suggestions and comments about my work.

This thesis was supported by the Walloon Region and Samtech s.a., in the frame of a FIRST Doctorat project called MINOS (*Moteur d’INférence pour l’Optimisation de Structures* – Inference Engine for Structural Optimization) under grant n° 001/4442. I would like to thank specially Dr. Claudine Bon (Samtech) for supporting the project, and Alain Remouchamps (Samtech) for helping me to make myself more familiar with Boss Quattro in particular, and optimization in general. Additional technical advice from Jérôme Coloos (Samtech) has also been deeply appreciated.

At the Université Libre de Bruxelles, my thanks go to Pr. Guy Warzée for his welcome in the Structural and Material Computational mechanics departement. I am also very thankful to Pr. Alain Delchambre, for his support as well as his important remarks about multicriteria design optimization, and Dr. Pierre De Lit for his help concerning the PROMETHEE II method.

I want to show my gratitude to Pr. Patrick De Wilde (Vrije Universiteit Brussel) and Pr. Pierre Villon (Université de Technologie de Compiègne) for supporting my project, as well as Pr. Piotr Breitkopf (UT Compiègne) for his help about genetic algorithms. Special thanks are also aimed at Dr. Joshua Knowles (IRIDIA/ULB), for his paramount comments about multiobjective optimization in evolutionary algorithms. I am also grateful to Vincent Kelner (ULg) for giving me interesting bibliographic references.

In order to build a method applicable to real-life cases, Techspace Aero engineers provided me much information and support. My thanks go first to Alain Navez, Mark Nott and Philippe Nomérange for their welcome and support in the company, and to Pierre Lassarrade and Michel Saint-Mard who worked on the VINCI engine project and made me familiar with space valves. Special thanks are also dedicated to Michel Pierson, Christophe Promper, James Murray, Carine Meurant, Daniel Renson, Albert Cornet, Luc Damhaut, Marc Noël, Frédéric Madon and all the other people (engineers and technicians) who were kind enough to spend some of their time to help me understanding the valve design.

May all the (former and current) members of the Structural and Material Computational mechanics dpt. who have not been cited yet be thanked for the very nice atmosphere that reigns in this department, namely : Adrian, Arnaud, Cristina, David, Elisabeth, Erik, Frédéric, Geneviève, Guy, Katy, Kfir, Laurent, Matthieu, Nadine, Nathalie, Olga, Raphaël, Tanguy, Thierry, Valérie, Valéry and Vincent. Finally, I would like to thank all my relatives and friends who kindly supported me during my PhD.

CHAPTER 1 – INTRODUCTION

1.1 Introduction to structural optimization

Since the 1960's, the design of mechanical components has been greatly enhanced thanks to the development of numerical methods. Finite element softwares, for instance, are now of common use in aeronautical, mechanical, naval and civil engineering. At the same time, efficient and fast optimization algorithms have arisen for solving various kinds of mathematical programming problems. Both trends gave birth to structural optimization, which aims at finding the best-fitted structure by modifying geometrical, material and/or topological parameters (the variables), the optimal solution being defined with respect to at least one criterion (the objective), and having to satisfy a set of requirements (the constraints) [PAP00].

In this work, the emphasis is mainly put on the optimization of mechanical components during the first stage of the design process, and it will be assumed that low cost reliable models are used (of course, this matter will be discussed thoroughly for the industrial applications studied at the end of the thesis). In fact, whereas lots of structural optimization techniques have been developed to increase the performances at the end of the design process, fewer works hitherto are concerned with optimization during the first stage, though a small modification in the beginning can bring significant improvements to the final structure.

Therefore, before presenting the objectives (§ 1.2) and an overview (§ 1.3) of the thesis, design optimization is first replaced in the general context of structural optimization.

Structural optimization is traditionally classified in three families following the nature of the variables involved [BRA86] (cf Fig. 1.1) :

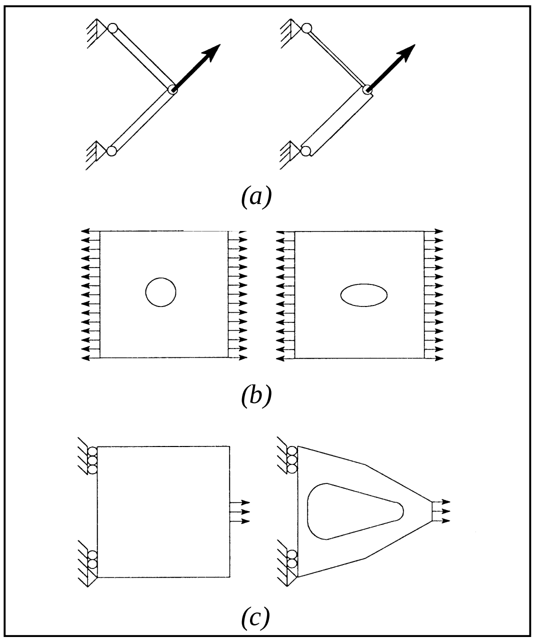


Fig. 1.1 : Distinction between design (a), shape (b) and topology (c) optimization [DUY96].

- in *design* or *sizing optimization*, variables represent only cross-sectional dimensions or transversal thicknesses (the geometry and the topology remaining fixed) ;
- in *shape optimization*, the variables are parameters acting directly on the geometry of the structure (but with a fixed topology) ;
- finally, *topological optimization* handles variables which can modify the shape and the topology of the structure.

These categories are briefly illustrated below.

1.1.1 Design optimization

In the first approach – also known as "automatic dimensioning of structures" [DUY96] –, the only variables are cross-sectional dimensions or transversal thicknesses (the geometry and the topology remaining fixed). In trusses for instance, the areas of the cross sections of the rods play the role of design variables, while the objective is commonly to find the lightest structure which still satisfies a set of constraints (e.g. stresses and displacements must not overstep maximum levels : see [AZI02,GRO99]).

1.1.2 Shape optimization

In shape optimization, the variables are geometrical parameters defining the shape of the structure (the topology remaining fixed). In most works available in the literature, the parameters are the coordinates of specific points : the poles. In 2D, these poles define the contour of the structure as a set of curves, for instance by using Lagrangian, Bézier or B-splines interpolations [AFO02,BRA84,ZHA92].

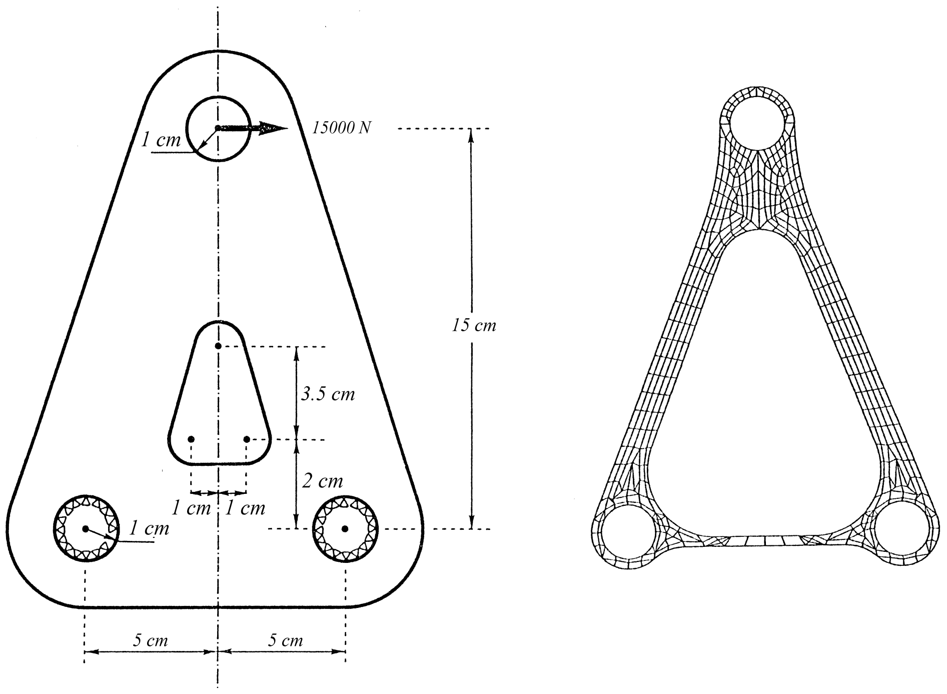


Fig. 1.2 : Shape optimization of a support : definition of the initial geometry (left) and solution obtained after 5 iterations of the optimization process (right) [ZHA92].

The geometry can also be modelled directly via lengths of segments, radii, angles, etc., considered thus as the design variables. This technique was illustrated by Zhang in the case of a support structure [ZHA92] : the objective was to minimize the mass without allowing the Von Mises stress to overstep a critical limit. The load case and boundary conditions are detailed in Fig. 1.2 (left). 10 independent variables are used, modelling the support geometry through lengths and arcs of circles. The structural analysis is performed thanks to a finite element model, and the optimal solution found by Zhang is represented in Fig. 1.2 (right), illustrating a mass decrease of 69.6% in comparison with the initial structure. To perform the optimization, CONLIN algorithm – based on the construction of convex linear approximations of the objective and constraints – was used for the computations.

1.1.3 Topological optimization

In topological optimization, the aim is to determine the optimal shape of a structure by starting with a bulk of material, and progressively taking off the material which undergo less loadings. Of course, the final structure must still satisfy the user-defined constraints (generally related to the restriction of the maximum Von Mises stress) [DUY96,NAK01,KIM02].

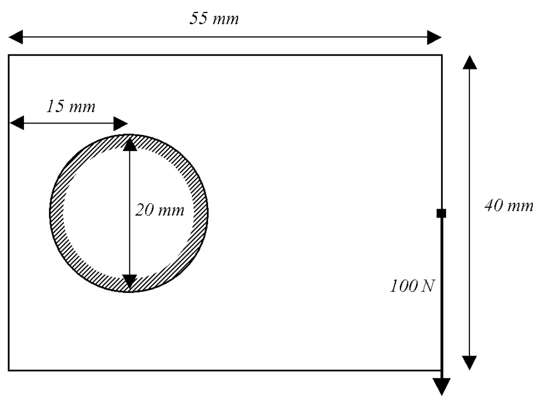


Fig. 1.3 : Definition of the Michell truss problem [Boundary condition : the inner circular hole is fixed] (figure adapted from [REY99]).

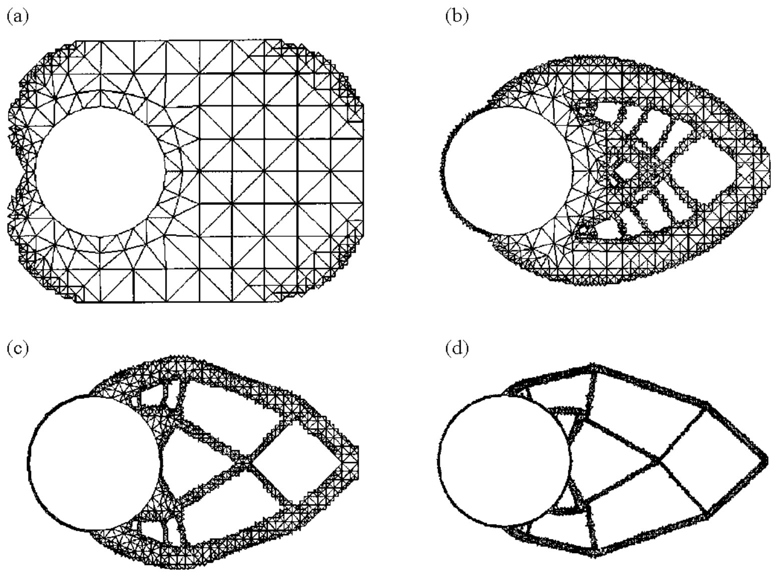


Fig. 1.4 : Topological optimization applied to the Michell truss problem : results at iterations 6 (a), 42 (b), 75 (c) and 120 (d) [REY99].

A classical benchmark of topological optimization, the Michell truss problem, is described in Fig. 1.3. Reynolds *et al.* solved it with the reverse adaptivity technique [REY99], which works as follows : once the initial finite element problem is built, the method proceeds with a refinement of low (Von Mises) stress regions of the mesh by element subdivision. Then, low stress subdivided elements are removed and the process is repeated. The structures obtained after respectively 6, 42, 75 and 120 iterations of this process are represented in Fig. 1.4. At the 120th iteration, only 8.8% of the whole (initial) area remains.

Topological optimization can also be achieved for trusses. For instance, Deb and Gulati described a method to find optimal cross-sectional areas and topology of 2-D and 3-D trusses by using genetic algorithms [DEB01a]. Topological variables representing the presence (or not) of each element in the configuration were introduced in addition to the design variables (i.e. the areas of the rod sections). The goal was to minimize the mass, while the stresses and displacements had to stand within allowable values. Figure 1.5 shows a 3D example where the genetic algorithm converged to a 9-element truss (from a 39-element initial configuration).

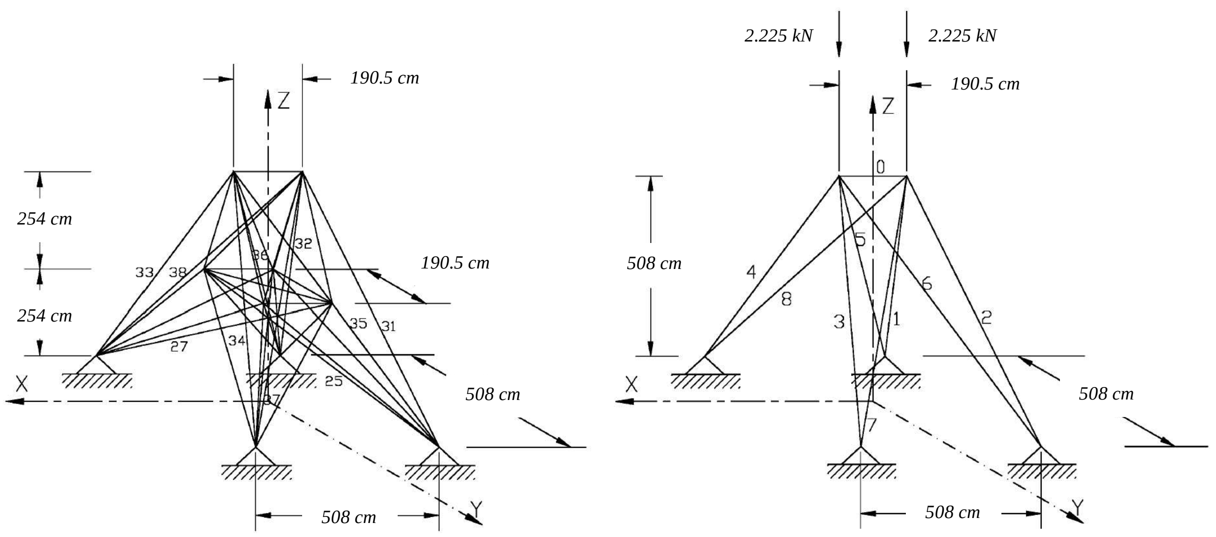


Fig. 1.5 : Example of truss topological optimization : from a network of 39 rods (left), the genetic algorithm converged to a 9-element truss (right) [DEB01a].

Latest developments in shape and topological optimization combined with finite element and boundary element methods are collected in [MAC02].

1.1.4 Other issues

Structural optimization can be linked to other fields of computational mechanics, like error estimation for example [ODE03]. Indeed, when an optimization algorithm interacts with a structural model, it modifies the structure geometry at each iteration and re-computes it. This raises the question of the model reliability (e.g. when the initial finite element mesh is perturbated due to the modification of the geometry). In some applications, it can be taken into acount directly in the optimization scheme. For instance, in [LAC03], Lacroix and Bouillard use a coupled finite element – element-free Galerkin (FE – EFG) method to improve the sensitivity analysis used in the optimization process, avoiding thus a mesh degradation due to the evolution of the geometry at each iteration.

Beside the model verification, another important aspect in structural optimization is related to the computational time. When the cost of the numerical model is high, it is profitable to use approximation methods. For example, in [VAN01], Vande Weyer used a mid-range design of experiments technique to build response surfaces, based upon the calculations of a set of points judiciously selected in the variable space [GOU99]. Then, instead of the complex model, the approximate one is optimized.

1.1.5 Topic of this work : pre-design optimization

The traditional subdivision in three families (design, shape and topology optimization) has become somewhat rigid. Indeed, during the first stage of the design for instance, the optimization of a preliminary sketch could involve not only cross-sectional and transversal variables, but also geometrical, or even topological ones that would have been parameterized (as the number of holes in a mechanical part). That could be considered as an intermediate method, akin to design, shape and topology optimization.

In this context, design optimization has become synonymous with finding the optimal dimensions of parameterized structures [OSY02] (where the parameters are not only cross-sectional or transversal variables). There are examples of design optimization in various fields of engineering ; here are a few applications treated by optimization methods :

- concentric springs [OSY02] ;
- electromagnetic systems [WIN95] ;
- pressure vessels [COE02] ;
- welded beams [COE02] ;
- reinforced concrete beams [SHI00] ;
- steel frames with semi-rigid connections [KAM01].

Once the design optimization problem is mathematically formulated (in terms of objective(s) to improve and constraints to fulfil), it can be solved by an appropriate optimization algorithm : this will be discussed thoroughly in Chapter 2.

Consequently, the topic concerned in this work, namely the optimization at the first stage of the design procedure, will be referred to as “pre-design optimization” throughout this thesis : it will mean (implicitly) that geometrical, but also material and topological variables may be taken into account in the optimization scheme.

1.2 Objectives of this work

As mentioned above, in this work, automation and optimization of pre-designs are investigated.

1.2.1 Two approaches

To perform this task, two different approaches are encountered in the literature :

- *optimization algorithms* : after the parameterization of the problem, optimization algorithms may be used. They act like “black-box” algorithms in the sense that they do not use any particular knowledge of the problem, except mathematical information (as the derivatives of the function in gradient-based algorithms for instance). An overview of optimization algorithms is presented in Chapter 2 ;

- *expert systems* : based on a set of rules, they include specific knowledge about a particular problem, collected among experts in a scientific field, and are therefore restricted to a few applications (see Chapter 5).

The scope of this thesis is to propose a novel method, which would take benefit of both approaches : being sufficiently general to be used for a large family of applications, it should still be able to incorporate specific knowledge about the problem involved.

1.2.2 Development of an original optimization method in two steps

Mechanical components can be divided in two categories : the simple parts (as screw bolts, joints, etc.) and the more complicated structures. In the former components, when the first sketches are devised "from scratch", design and optimization are inextricably bound, whilst in the latter, the two stages can be distinctly separated. Fruitful discussion with engineers from Samtech s.a. and Techspace Aero led to the conclusion that the development of an optimization method applicable to complex structures (as valves, pomps, etc.) should thus be divided in two steps :

1. first, developing a tool to optimize the parametrical design of structures whose topologies are already fixed. The idea lying behind this approach is very simple : to compare two different designs (i.e. with different topologies), one has to optimize both parameterized designs following the same criteria, to prevent from having an intrinsically "good" design ill optimized outperformed by a "worse" design correctly optimized ;

2. once the first step is accomplished, proposing a method to optimize a more general design, modelled by geometrical, material and also topological variables. As different designs will be analyzed and inspected simultaneously, one way to reduce the search space and furnish a realistic solution is to incorporate expert rules (taking technological aspects into account) within the optimization process.

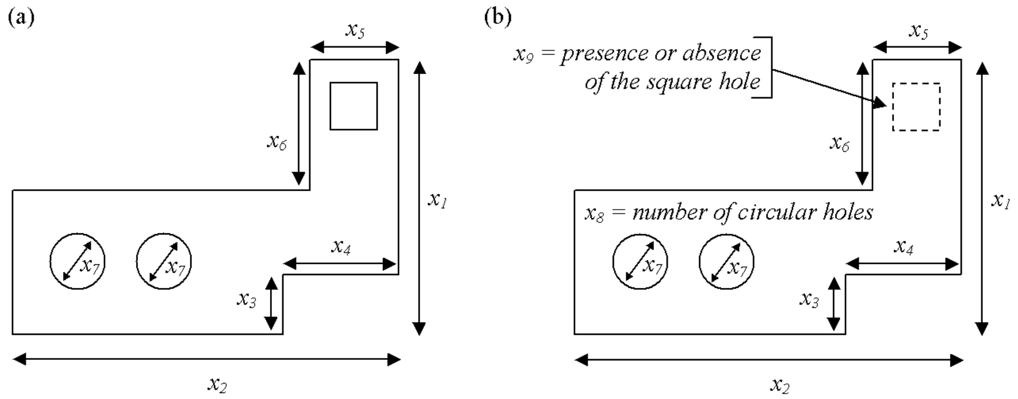

(a)
(b)
x_5
x_6
x_1
x_4
x_3
x_2
x_7
x_9 = presence or absence of the square hole
x_8 = number of circular holes

Fig. 1.6 : Examples of sheer design optimization (i.e. with only "dimensional" variables) (a) and of design optimization with topological variables (b).

Both steps are illustrated in Fig. 1.6. The final goal is therefore to achieve design optimization with topological variables. To perform this task, the method proposed in this thesis should succeed in finding optimal but also realistic solutions (in terms of technological requirements). As only the initial step of the design procedure is concerned, it is assumed that the objective function(s) and constraints have a low computational time, which seems a reasonable hypothesis since they are generally issued from theoretical and experimental models.

1.3 Overview of the thesis

The thesis is divided in two parts : first, performing parametrical pre-design optimization, and then incorporating expert rules to optimize more general pre-designs.

Once the design is parameterized, a suitable optimization algorithm has to be selected. This is discussed in Chapter 2, where a classification of the main optimization methods is performed, followed by the reasons of using evolutionary algorithms (EAs) for pre-design optimization. The features of the standard EA are also described, and some applications in structural optimization are mentioned.

Then, since most industrial applications deal with multiple objectives and strong technical requirements, the standard EA has to be adapted in order to tackle both multicriteria and constrained aspects. After a review of the main techniques used to take those aspects into account, underlining the lack of methods specially devoted to the simultaneous handling of preferences and constraints, an original approach is proposed, called PAMUC (*Preferences Applied to MUltiobjectivity and Constraints),* presented in Chapter 3.

Afterwards, a procedure to validate PAMUC is discussed. First, single-objective constrained problems are treated to show PAMUC efficiency to find feasible solutions. Secondly, multiobjective test cases are compared to the classical weighted sum method, thanks to a specific norm whose choice is debated thoroughly in Chapter 4.

Once PAMUC has been successfully validated for parametrical design, it can be extended to more general problems, i.e. dealing with topological variables. However, using a classical optimization method could lead to designs which would be optimal but unrealistic for technological reasons (e.g. related to the machining and the assembly). Therefore, to generate solutions satisfying also the technological constraints, one approach consists in incorporating expert rules within the algorithm. After a discussion of knowledge representation by logical rules (as in expert systems), and their possible insertion in an EA, an original method, called PAMUC II, is described, using the rules to repair unfeasible individuals among the members of the EA population. Then, it is applied on several test cases and mechanical benchmarks, and numerical aspects (e.g. the computational cost) are discussed (cf. Chapter 5).

Finally, PAMUC (II) is applied to four valves designed by Techspace Aero for launcher Ariane 5 to show its adequacy in solving real-life pre-design optimization problems (Chapter 6) ; general conclusions are drawn in Chapter 7.

CHAPTER 2 – EVOLUTIONARY ALGORITHMS APPLIED TO MECHANICAL DESIGN OPTIMIZATION

2.1 Introduction

Once a mechanical pre-design is correctly parameterized – with a fixed topology – an efficient optimization method has to be chosen. This constitutes the first step of the thesis.

Several approaches are available in the literature to handle optimization problems. Therefore, this chapter is divided in two parts : first, the different criteria upon which optimization problems are classified, as well as a brief overview of the main families of algorithms, are exposed (§ 2.2). Then, the emphasis is put on evolutionary algorithms (§ 2.3) : the reasons why they were selected, their working and some applications in engineering design will be discussed.

2.2 Classification of optimization problems and methods

A general optimization problem can be written as follows [FON95] :

$$\min_{x} \mathbf{f}(\mathbf{x}) \tag{2.1}$$

$$s.t.: \begin{cases} \mathbf{g}(\mathbf{x}) \geq 0, & (2.2) \\ \mathbf{h}(\mathbf{x}) = 0, & (2.3) \\ x_i \in X_i \;\; for\ i = 1,\ldots, n, & (2.4) \end{cases}$$

where :

- $\mathbf{x}^T = \{ x_1 \;\; x_2 \ldots x_n \}$ (vector of variables) ;
- X_i is the set of x_i (which may be continuous, discrete or integer) ;
- $\mathbf{f}(\mathbf{x})^T = \{ f_1(\mathbf{x}) \;\; f_2(\mathbf{x}) \ldots f_m(\mathbf{x}) \}$ (objectives) ;
- $\mathbf{g}(\mathbf{x})^T = \{ g_1(\mathbf{x}) \;\; g_2(\mathbf{x}) \ldots g_p(\mathbf{x}) \}$ (inequality constraints) ;
- $\mathbf{h}(\mathbf{x})^T = \{ h_1(\mathbf{x}) \;\; h_2(\mathbf{x}) \ldots h_q(\mathbf{x}) \}$ (equality constraints).

Note that henceforward bold letters represent vectors or matrices while plain letters stand for scalars. Optimization problems are classified upon various criteria :

- the nature of the variable sets : a variable may be continuous (e.g. a geometrical dimension), discrete (e.g. cross sections of beams are often available by discrete steps in catalogues) or integer (e.g. the number of layers in a composite material [SOR01]). There are often mixed variables in engineering problems (cf. [CAO00, COS01, SEN96, SHI00]) ;

- the nature of the constraints and the objective functions : they may be linear, quadratic, non linear or even non differentiable. For instance, gradient-based algorithms, based on the computation of the sensitivities, require the functions to be differentiable in order to compute their first-order (and sometimes also their second-order) derivatives [FLE78,WRI99] ;

- the analytical properties of the functions, e.g. linearity in linear programming. Convexity or monotonicity can also be successfully exploited to converge to a global optimal solution [PAR00] ;

- the presence (or the absence) of constraints. Equality constraints are usually tackled by converting them into inequality constraints [DEB00], using Eq. (2.5) (where the parameters ε_j are chosen by the user) :

$$h_j(\mathbf{x}) = 0 \;\rightarrow\; \varepsilon_j - |\, h_j(\mathbf{x})| \geq 0 \quad for\; j = 1,\ldots, q \;; \tag{2.5}$$

- the size of the problem : to remain applicable even when the number of variables is very large (more than about 10,000 for continuous problems), optimization algorithms have to be adapted, because of limited memory or computational time [WRI99] ;

- implicit or explicit functions : in shape optimization for instance, when finite element models are needed to compute the stresses and displacements, the objective function (generally the mass) is almost always an implicit function of the variables. Therefore, the objective(s) and constraints are approximated thanks to a linear, quadratic or other (cubic, posynomial, etc.) model [FLE00,SAL00]. These approximations are usually built in order to exhibit specific properties, as convexity in the CONLIN method [REM99]. Neural networks may also be used to construct an approximation of the functions [RAF01] ;

- local or global optimization : in single-objective optimization, this distinction is based on the following definitions (written – without limitation – in the case of minimization) :

  - a point $\mathbf{x}^*$ is said to be a *global minimizer* if (and only if) $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in X$ (the whole design space) [WRI99] ;

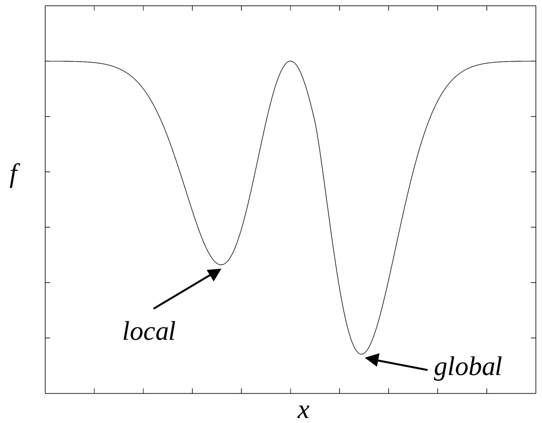


Fig. 2.1 : Example of a 1-variable function with the global minimum and one local minimum.

- a point $\boldsymbol{x}^*$ is said to be a *local minimizer* if there exists a neighbourhood $\aleph$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for $\boldsymbol{x} \in \aleph$. Of course a global minimum is also a local one.

  Local optimization is used commonly with smooth functions in order to find a local optimum. Note that when the functions involved are also convex, the local optimizer is also a global one. An example of a 1-variable function with the global minimum and one local minimum is illustrated in Fig. 2.1.

- single-objective or multiobjective : though the first studies in structural optimization used only one objective (most of time minimizing the mass), more and more studies deal with multiple criteria (mass, cost, specific performances, etc. ; see [AND00, COE96,DEB99b]). Indeed, in industrial context, optimal solutions must be good compromises between the different (and often contradictory) criteria (cf. [CVE00,MAS99, ZHA01a]).

To solve optimization problems, a huge amount of methods have been proposed in the literature. They are briefly summarized below.

2.2.1 Local methods

Local methods are aimed to reach a local optimum, and offer no guarantee in finding the global one. The most common local methods are based on the computation of sensitivities. Therefore, they require the functions to be differentiable, and the variables to be continuous (or discrete). Nocedal *et al.* divided the main gradient-based algorithms in two approaches [WRI99] :

- in the *line search* strategy, the algorithm chooses a direction $\boldsymbol{p}_k$ (e.g. the steepest-descent direction) and performs a search for a better point $\boldsymbol{x}_k$ along this direction (see Fig. 2.2). Each iteration needs to solve the following one-dimensional minimization problem :

$$\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k). \tag{2.6}$$

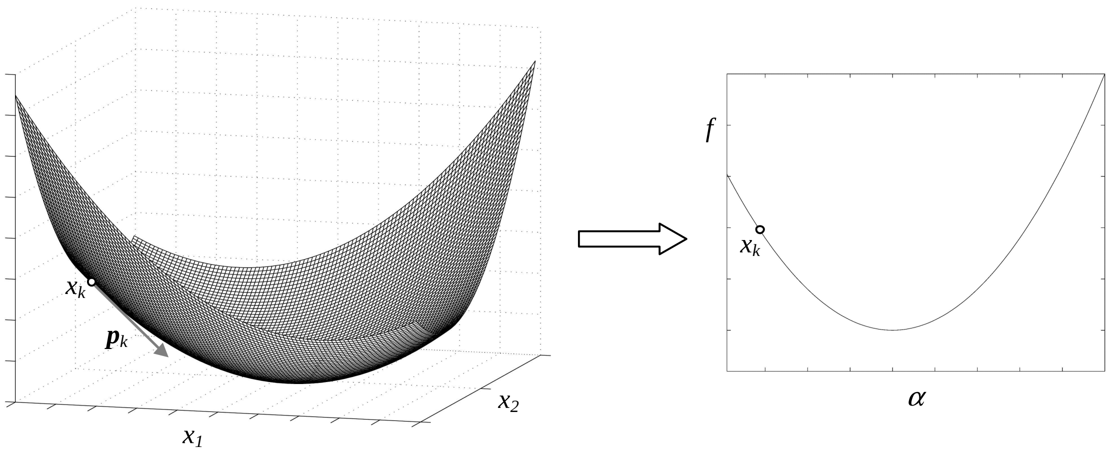


Fig. 2.2 : Principle of the line search strategy (with a 2-variable function f) : once a direction $\boldsymbol{p}_k$ *has been chosen, a 1-dimensional minimization is performed along that direction (figure adapted from [WRI99]).*

- in *trust region* methods, the gradient $\nabla f(\boldsymbol{x}_k)$ at iteration k (and sometimes also the Hessian matrix $\boldsymbol{B}(\boldsymbol{x}_k)$, i.e. the second derivatives) is used to construct a model m_k whose behaviour is a good approximation of the original function f, at least in a close neighbourhood of $\boldsymbol{x}_k$ (cf. Fig. 2.3). Each iteration consists in solving the sub-problem :

$$\min_{\boldsymbol{p}} \; m_k(\boldsymbol{x}_k + \boldsymbol{p}), \tag{2.7}$$

where $\boldsymbol{x}_k + \boldsymbol{p}$ lies inside the trust region. Usually the model m_k is defined as a quadratic function of the form :

$$m_k(\boldsymbol{x}_k + \boldsymbol{p}) = f(\boldsymbol{x}_k) + \boldsymbol{p}^T \nabla f(\boldsymbol{x}_k) + ½\, \boldsymbol{p}^T \boldsymbol{B}(\boldsymbol{x}_k) \boldsymbol{p}. \tag{2.8}$$

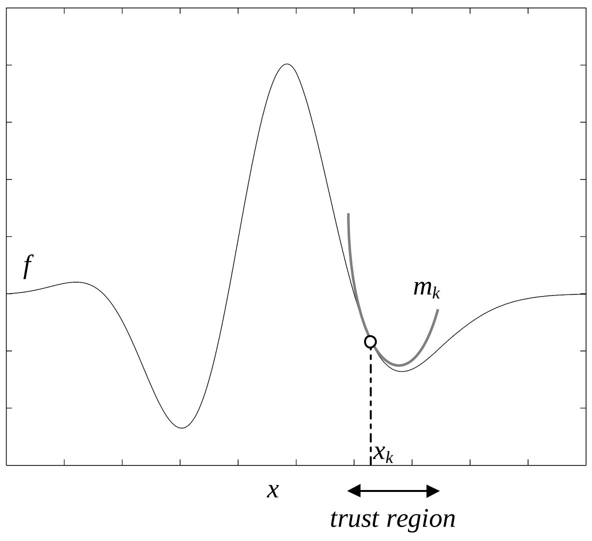


Fig. 2.3 : Principle of the trust region strategy (with a 1-variable function f) : at each iteration, an approximated model m_k is constructed in the vicinity of x_k (and reliable within the trust region) ; then, m_k is minimized instead of the true objective function f (figure adapted from [WRI99]).

Various instances of these techniques have been proposed in the literature [WRI99] : the BFGS algorithm, the conjugated gradient method, the sequential quadratic programming (SQP), etc. Though these algorithms were initially restricted to continuous unconstrained problems, they have been successfully extended to other fields of optimization :

- *constrained optimization* : Lagrange multipliers allow the user to take constraints (equalities and inequalities) into account [CIA98] ;

- *discrete optimization* : in this approach, the problem is solved in a dual space to deal with discrete variables [BEC00,HUA97,SEP86]. Some interesting applications in structural optimization (sizing of thin-walled structures, geometrical configuration of trusses, topological optimization of membranes or 3-D structures, etc.) are mentioned in [BEC00] ;

- *integer or combinatorial programming* : those problems can often be transformed in constrained continuous problems, for example by reformulating the "binary" constraints as follows :

$$x_i = \{0,1\} \Leftrightarrow x_i.(x_i - 1) \geq 0 \ , \ 0 \leq x_i \leq 1 \ \ \forall i = 1,\ldots, n. \tag{2.9}$$

Unfortunately, as mentioned above, all algorithms based on the classical gradient techniques require the functions to be differentiable, which is often not the case in design optimization problems. Furthermore, even if the functions were differentiable, the risk is high to be trapped in a local minimum. Therefore, global methods seem more suited to solve general design optimization problems. The main global approaches are summarized in the next section.

2.2.2 Global methods

Reaching the global optimum is an arduous task, which explains that various techniques have been proposed to handle it [MAN99,VIS90,YOU01]. The most popular ones are briefly described hereafter :

- *Random search* : in this basic (and time-consuming) technique, a large number of points $\boldsymbol{x}_1, \boldsymbol{x}_2,\ldots, \boldsymbol{x}_N$ are randomly generated and their corresponding function values $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2),\ldots, f(\boldsymbol{x}_N)$ computed ; the point $\boldsymbol{x}^*$ endowed with the best function value is selected to be the solution [OZD00] ;

- *Approximation methods* : instead of searching directly the optimum of the true function, an approximated function is built in order to solve the problem more easily. This approximation can be a statistical function, or a response surface built upon a set of function values computed for a predefined sample of variables (by the design of experiments technique [GOU99,VAN01]). Neural network methods may also be used to approximate the objective function(s) and constraints [HUR01,RAF01] ;

- *Clustering methods* : they can be viewed as a modified form of the standard multistart procedure, which performs a local search from several points distributed over the entire search domain. A drawback of pure multistart is that when many starting points are used, the same local minimum may be obtained several times, thereby leading to an inefficient global search. Clustering methods attempt to avoid this inefficiency by carefully selecting points at which the local search is initiated [FAS99] ;

- *Metaheuristic methods* : from the definitions collected in three dictionaries specialized in mathematical programming, algorithms and data structures [GRE03, HOW93,NIS03], metaheuristics can be defined as general strategies which guide the search for optimal solutions in hard problems. In artificial intelligence, *heuristic methods* are non deterministic algorithms based on a set of rules of thumb that a human would make following his/her intuition ; therefore, no guarantee is offered that the optimal solution will be found systematically. The prefix *meta-* is related to the level of abstraction of these techniques.

  In evolutionary algorithms for example, it is implicitly assumed (but not proved) that when good potential solutions are matched, they generally produce better solutions. Besides, their use is not restricted to a narrow family of applications, but to a large class of problems [YAG96] ; hence they belong to *metaheuristics*.

Additionally, lots of metaheuristics are inspired by an analogy with physical or biological processes. Some of the most widespread metaheuristics in structural optimization are recapitulated below :

- *Simulated annealing* : this method derives from the annealing process in metallurgy, which consists of first raising the temperature *T* to a point where the atoms can freely move, and then to lower *T* to force the atoms to rearrange themselves into a lower energy state (crystallisation) [LAR88]. During this process, the free energy of the solid is minimized. By associating the objective function of an optimization problem to the free energy of a material, an efficient search procedure has emerged. The cooling schedule is a crucial parameter of simulated annealing : indeed, if the solid is cooled too quickly, or if the initial temperature is too low, the solid will reach an amorphous state instead of becoming a crystal. In the language of programming, it means that the algorithm has converged to a local minimum [COE02a].

  The basic simulated annealing algorithm functions as follows : a small perturbation is applied to the starting point, which modifies its energy (i.e. its objective function) : if the change is negative (in a minimization problem), the new configuration is better than the original, and is therefore accepted ; but if it were positive, the new point would still be accepted with a probability *P* given by Eq. (2.10) :

  $$P = exp(-\Delta E/kT), \tag{2.10}$$

  where *k* is the Boltzmann constant, relating the temperature *T* to the variation of energy ΔE [LAR88]. This allows the algorithm to explore regions located outside the vicinity of local minima ;

- *Tabu search* : in this technique, at each iteration, a feasible move is applied to the current point, accepting the neighbour with the smallest cost. Tabu search acts like a local search method, except that positions which seem not favourable may be allowed to prevent from converging to the same (maybe local) optimum [BLA98,FRA01]. Tabu search also forbids reverse moves to avoid cycling (the forbidden movements are “quarantined” and compose the so-called tabu list [MAC01, VOU99]) ;

- *Ant colony systems* : they are based on the behaviour of real ants, observed by Gross *et al.* in laboratory in the following experience (cited in [DOR99]) : a colony of Argentine ants were given access to a food source in an arena linked to the colony nest by a bridge with two branches of different lengths, disposed in such a way that ants looking for food must choose between either branch or the other. After a transitory phase, most of the ants took the shorter branch. This can be explained by the fact that ants, while going from nest to food and vice versa, deposit a chemical substance (the pheromone) along the way. When an ant has to choose between two paths, it will instinctively be attracted by the path containing the larger amount of pheromone, and as the shorter branch is the smaller distance for the ants, they make the trip faster and therefore more pheromone is released there.

The ant colony optimization algorithm is roughly based upon the observations made in this experience, but with several adaptations following the problem to be solved [DOR99] ;

- *Evolutionary algorithms (EAs)* : they were originally separated into three families : genetic algorithms [GOL89], evolution strategies [BAC91,BAC93,MOR99] and genetic programming [YAN02]. Now, however, these techniques become closer and closer [COE99], and differ mainly in the coding of the variables and the relative importance given to each genetic operator. These methods are search techniques which follow the Darwinian law of natural selection or survival of the fittest [BAC92,BAC96,GOL89] : in a random population of potential solutions, the best individuals are favoured and combined in order to create better individuals at the next generation. The implementation of the standard EA, as well as theoretical aspects and applications, will be discussed in the next section.

As some identical procedures exist in different metaheuristic methods (e.g. the need of keeping in memory some data about previous results), Taillard *et al.* proposed the use of the generic term *adaptive memory programming* to label the different families of metaheuristics mentioned above [TAI98]. Another aspect which is common in all metaheuristics is the fact that they can be successfully combined to gradient-based algorithms to construct hybrid methods (cf. e.g. [ALO01, DUL02]). In EAs for instance, a local search may be applied to improve the initial generation as well as the individuals obtained by recombination, to form the *memetic algorithms* [WIN95].

The choice of EAs for design optimization is discussed in the next section, followed by their implementation and some examples of applications in structural optimization.

2.3 Evolutionary algorithms for design optimization

2.3.1 Why choosing evolutionary algorithms ?

For design optimization, the use of evolutionary algorithms seems very attractive, for the following reasons :

- *the nature of the variables* : as mentioned in Chapter 1, design optimization problems may need the use of mixed variables (continuous, discrete and/or integer). This can be handled very easily in EAs [DEB98,REI97,ULU02], whereas gradient-based algorithms for instance are mainly devoted to problems with continuous variables ; discrete variables may also be used, but only in some specific applications [BEC00] ;

- *the nature of the functions* : as the functions (empirical formulas for example) involved in design optimization are often non differentiable, and sometimes discontinuous, gradient-based techniques are once more excluded. Only 0-order algorithms systematically fit, because they require only the values of the functions, and not their derivatives ;

- *exploration of the search space* : as they work on a population of solutions instead of a single point (at each iteration), EAs are less likely to be trapped in a local minimum. Goldberg also proved thanks to the schemata theorem [GOL89] that the way recombination of individuals is performed allows the algorithm to explore widely the whole design space. EAs are thus very well suited for noisy and multimodal functions [KAL01].

Therefore, EAs are ideally adapted for structural optimization, as can be seen by the impressing number of applications in engineering (see [BUR02] and other examples in § 2.3.3).

2.3.2 Description of the standard EA

The standard evolutionary algorithm (Std-EA) mentioned throughout this thesis is a genetic algorithm whose working is described in [MIC96a,BAC97]. It was implemented by the author in Matlab, and without additional procedures, it is aimed to solve single-objective unconstrained problems. Its basic features are briefly summarized below.

2.3.2.1 Flow-chart of the algorithm

The flow-chart of the standard evolutionary algorithm is represented in Fig. 2.4.

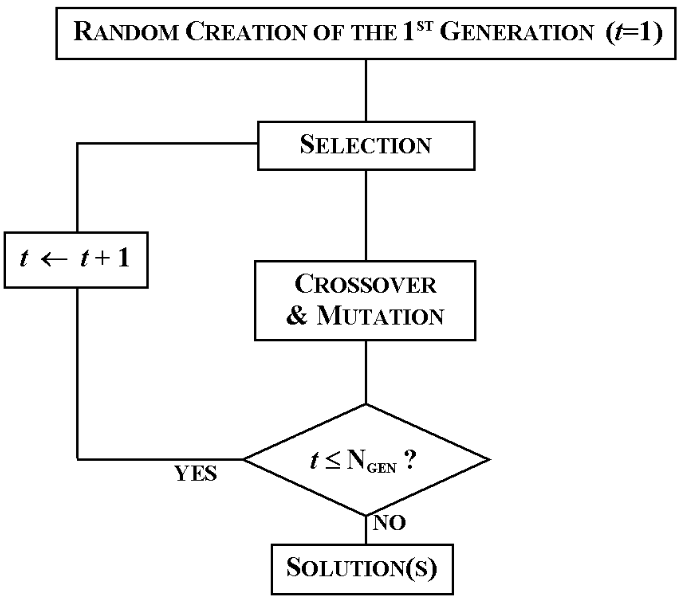


Fig. 2.4 : Flow-chart of the standard evolutionary algorithm (figure adapted from [MIC96a]).

After the random generation of the initial population, the individuals are selected following the value of their *fitness function* : the individuals with highest fitness values are more likely to be chosen to take part of the process of recombination. In the case of an unconstrained optimization problem, the objective generally plays the role of the fitness function ; otherwise, a more sophisticated function has to be constructed to reflect correctly the quality of an individual (e.g. in constrained optimization, a solution with a poor value of the objective function but satisfying the constraints is often preferred to a unfeasible one endowed with a a better objective function).

2.3.2.2 *Coding of the variables*

A crucial point in evolutionary algorithms lies in the coding of the variables. The variables of each individual of the population are coded in a *chromosome*. By analogy with genetics, the values of the variables are called the *phenotype* and the coding the *genotype*. Four different codings have been implemented in the Std-EA :

1. **a (classical) binary coding** [BAC97] : each variable is coded in a substring of bits whose number is related to the number of *alleles* (i.e. possible values) that the variable could take. This is illustrated in the case of mixed variables in Table 2.1.

Variables	Type of variable	X_i (Variation domains of x_i)	Number of alleles	Substring size
x_1	Continuous	[0,10]	1024 (= 2^{10})	10
x_2	Continuous	[0,10]	1024 (= 2^{10})	10
x_3	Discrete	{10 ; 12.5 ; 15 ; 17.5 }	4 (= 2^2)	2
x_4	Integer	{0 ; 1}	2 (= 2^1)	1

Table 2.1 : Example of binary coding : construction of a chromosome (4 design variables) [BAC97].

The chromosome of an individual is then constructed by concatenating the substrings S_i corresponding to each variable x_i (cf. Fig. 2.5) :

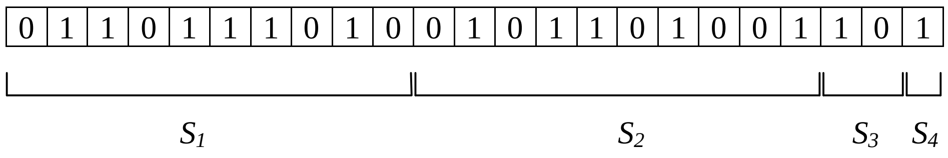


Fig. 2.5 : Example of chromosome for a four-variable individual (with binary coding) [BAC97].

2. **a Gray binary coding** : the binary representation as described above is widely used in the EA community, but it has some drawbacks. Indeed, it is commonly accepted that a coding should reflect as closely as possible the behaviour of the variables. For example, a small change in the value of the variable should lead to a small modification of the genotype. This is not systematically the case in binary coding, where subsequent alleles may have completely different chromosomes. Therefore, the Gray coding has been introduced, and is built in such a way that two subsequent alleles differ only from one bit [OSY02] (cf. Table 2.2 for a 3-bit variable).

Allele	Binary coding	Gray coding
1	000	000
2	001	001
3	010	011
4	011	010
5	100	110
6	101	111
7	110	101
8	111	100

Table 2.2 : Binary and Gray codings for a 3-bit variable [OSY02].

3. **a fixed-point representation** [BAC97] : this coding is based on a decimal representation : each division of the chromosome corresponds to one figure, and the place of the decimal point is fixed. This is illustrated in Fig. 2.6 for a 2-variable individual.

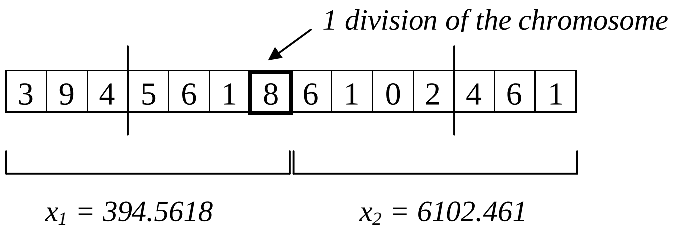


Fig. 2.6 : Fixed-point representation for a 2-variable individual.

4. **a real coding** : when there are only continuous variables, a real coding is often preferred, because it is very close to the real search space [ANI02,OSY02]. In this representation, each individual is thus coded as a vector of real values.

Though it has not been implemented in the Std-EA, it is also possible to have a mixed chromosome representation, combining different codings of the variables, as in the Genetic Adaptive Search method developed by Deb [DEB98]. Other representations are possible, as long as they precisely describe the problem ; guidelines to build a suitable coding are available in [BAC97].

2.3.2.3 Creation of the initial population

The first step of the Std-EA consists in generating randomly the initial population of N individuals. N is a user-defined parameter called the size of the population.

2.3.2.4 Selection

After the evaluation of the fitness function for each individual of the population, the selection is performed. Two classical selection schemes were implemented in the Std-EA [BAC97] :

1. **the roulette wheel selection** : in this procedure, each chromosome has a given probability of selection, which is a (monotonous) function of its fitness. In Fig. 2.7, the roulette wheel is symbolically represented for 5 individuals i_1 to i_5 : each slot of the wheel has a size proportional to the probability – for the corresponding individual – of being selected.

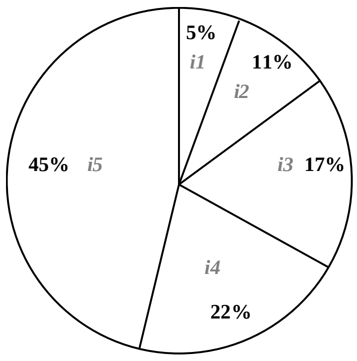


Fig. 2.7 : Roulette wheel selection
(example for five individuals ; the percentages indicate the probability of selection of the individuals).

2. **the tournament selection** : n_t individuals (the *knights*) are randomly chosen among the previous generation (with $n_t \leq N$), and compared following their fitness values. Then, the best individual is copied in the new generation [MIL95].

The selection process is repeated until N individuals have been chosen. This procedure is called *selection with replacement* [BAC97].

2.3.2.5 Crossover

Once the individuals have been selected, they are divided in $N/2$ pairs of *parents*, and matched – with a user-defined probability – by the crossover procedure. In binary, Gray and fixed-point codings, three instances of crossovers were implemented in the Std-EA [HAS00] :

1. **the 1-point crossover** : two *children* are constructed by inverting the genes of their *parents* from the (randomly determined) crossover site (see Fig. 2.8) :

PARENT 1	1	0	1	0	1	0	0	1	1	0	1	1	1	0	1	0	1	0	1	0	1	1	0
PARENT 2	***1***	***1***	***0***	***0***	***0***	***1***	***0***	***0***	***1***	***1***	***0***	***1***	***0***	***0***	***1***	***0***	***0***	***1***	***1***	***0***	***1***	***0***	***1***

crossover site

CHILD 1	1	0	1	0	1	0	0	1	***1***	***1***	***0***	***1***	***0***	***0***	***1***	***0***	***0***	***1***	***1***	***0***	***1***	***0***	***1***
CHILD 2	***1***	***1***	***0***	***0***	***0***	***1***	***0***	***0***	1	0	1	1	1	0	1	0	1	0	1	0	1	1	0

Fig. 2.8 : Illustration of the 1-site crossover :
two strings (the parents) and their offspring (figure adapted from [HAS00]).

2. **the 2-point crossover** : the procedure is the same as in the 1-point crossover, except that 2 crossover sites are randomly chosen ;

3. **the uniform crossover** : first, a binary string is randomly generated. Then, two *children* are constructed by inverting the genes of their *parents* following the value of the corresponding bit in the random string (see Fig. 2.9) :

PARENT 1	1	0	1	0	1	0	0	1	1	0	1	1	1	0	1	0	1	0	1	0	1	1	0
PARENT 2	***1***	***1***	***0***	***0***	***0***	***1***	***0***	***0***	***1***	***1***	***0***	***1***	***0***	***0***	***1***	***0***	***0***	***1***	***1***	***0***	***1***	***0***	***1***

RANDOM STRING	1	0	1	0	0	1	0	1	1	0	0	1	0	1	1	1	1	0	1	1	0	1	0

CHILD 1	1	***1***	1	***0***	***0***	0	***0***	1	1	***1***	***0***	1	***0***	0	1	0	1	***1***	1	0	***1***	1	***1***
CHILD 2	***1***	0	***0***	0	1	***1***	0	***0***	***1***	0	1	***1***	1	***0***	***1***	***0***	***0***	0	***1***	***0***	1	***0***	0

Fig. 2.9 : Uniform crossover (for Child 1, a value of 1 in the random string corresponds to a bit from Parent 1, and 0 corresponds to a bit from Parent 2, and vice versa for Child 2) (figure adapted from [HAS00]).

For real coding, the SBX (simulated binary crossover), proposed by Deb [DEB95,DEB96], was incorporated in the Std-EA. To compute two children $y^{(1)}$ and $y^{(2)}$ from two parents $x^{(1)}$ and $x^{(2)}$, the following technique is applied :

- create a random number u between 0 and 1 ;
- compute the following parameter :

$$\overline{\beta} = \begin{cases} (\alpha u)^{1/(\eta_c+1)} & if\ u \leq \alpha^{-1}, \\ \left(\dfrac{1}{2-\alpha u}\right)^{1/(\eta_c+1)} & otherwise, \end{cases} \tag{2.11}$$

  where :

$$\alpha = 2 - \beta^{-(\eta_c+1)}, \tag{2.12}$$

$$\beta = 1 + \frac{2}{y^{(2)} - y^{(1)}} min\left[(x^{(1)} - x^l),(x^u - x^{(2)})\right]. \tag{2.13}$$

  In Eqs. (2.11) to (2.13), it is assumed that $x^{(1)} < x^{(2)}$; modifications of the above formulae are easily made for $x^{(1)} > x^{(2)}$. x^l and x^u are respectively the lower and upper bounds of the variable, and η_c is a user-defined parameter (standard value used in [DEB00] is $\eta_c = 1$).

- the children solutions are then computed as follows :

$$y^{(1)} = 0.5\ \left[(x^{(1)} + x^{(2)}) - \overline{\beta}\left|x^{(2)} - x^{(1)}\right|\right], \tag{2.14}$$

$$y^{(2)} = 0.5\ \left[(x^{(1)} + x^{(2)}) + \overline{\beta}\left|x^{(2)} - x^{(1)}\right|\right]. \tag{2.15}$$

2.3.2.6 Mutation

Mutation is a useful complement of crossover, for it enables to explore possibly undiscovered areas of the search space. Though more sophisticated mutation operators are mentioned in the literature [GUT99], for binary, Gray and fixed-point representation, only the classical “flip” mutation was implemented [BAC97] : with a user-defined probability, the value of one bit (or one division in decimal coding) is randomly changed.

In real coding, the parameter-based mutation operator is used [DEB95]. To compute the mutated solution, the following procedure is applied :

- create a random number u between 0 and 1 ;
- compute the following parameter :

$$\overline{\delta} = \begin{cases} -1 + \left[\, 2u + (1-2u)(1-\delta)^{\eta_m+1} \,\right]^{1/(\eta_m+1)} & if\ u \leq 0.5, \\ 1 - \left[\, 2(1-u) + 2(u-0.5)(1-\delta)^{\eta_m+1} \,\right]^{1/(\eta_m+1)} & otherwise, \end{cases} \tag{2.16}$$

where :

$$\delta = \frac{min\left[(x - x^l),(x^u - x) \right]}{x^u - x^l} \tag{2.17}$$

and η_m is the distribution index for mutation and takes any nonnegative value ;

- calculate the mutated child using Eq. (2.18) :

$$y = x + \bar{\delta}(x^u - x^l). \tag{2.18}$$

As in [DEB00], the value of η_m is computed as follows :

$$\eta_m = 100 + t, \tag{2.19}$$

where t is the current generation number. The mutation rate p_m is given by :

$$p_m = \frac{1}{n} + \frac{t}{t_{max}}\left(1 - \frac{1}{n}\right), \tag{2.20}$$

where t_{max} is the maximum number of generations allowed. More details about the SBX and parameter-based mutation operator can be found in [DEB95,DEB96].

After the operators of recombination (crossover and mutation) have been applied, for each pair of matched parents, the two best individuals among the four (i.e. among the two parents and the two children) are preserved to take part of the selection scheme ; this ensures the algorithm to keep the best individuals during the generations.

2.3.2.7 Remarks

Other genetic operators (as niching, elitism or inversion) exist, and various implementations have flourished since the late 1990's (see [COE00d,HIN97,KOU02,LEI98]), but the Std-EA implemented in this study, albeit simple, is sufficiently general and efficient to be adapted to multicriteria and constrained optimization, as well as to the introduction of expert rules, as it will be exhibited in the next chapters.

The different parameters of the Std-EA are summarized in Tables 2.3 and 2.4. It is important to notice that in EAs, the definition of the parameter values is crucial, and there is no systematic method yet to determine their optimal values : a tuning of these parameters has still to be done by the user for each application.

Parameter	**Definition or possible values**
N	Size of the population
N_{gen}	Maximum number of generations
Type of selection	Roulette wheel or tournament
n_t	Number of individuals participating to a tournament ($\leq N$) (when the tournament selection is performed)
p_c	Probability of crossover ($0 \leq p_c \leq 1$)

Table 2.3 : List of parameters of the Std-EA not depending on the coding.

Parameter	**Definition or possible values**	
Coding	***Binary, Gray or decimal coding***	***Real coding***
Type of crossover	1-site, 2-site or uniform	SBX
Type of mutation	Flip	Parameter-based mutation
p_m	Probability of mutation (defined by the user, with $0 \le p_m \le 1$)	Probability of mutation (function of the generation number t : cf. Eq. (2.20))
η_c	–	Distribution index for crossover (≥ 0)
η_m	–	Distribution index for mutation (≥ 0)

Table 2.4 : List of parameters of the Std-EA depending on the coding.

2.3.3 Miscellaneous examples of applications in engineering

The robustness of EAs has been exploited in a broad family of applications : this section presents a few examples of mechanical problems successfully treated with EAs :

- in [JEN97], the volume of a multistorey frame with truss-supported hangers is minimized by acting on geometrical variables (cross sections of beams and columns and lengths of structural elements), while constraints are defined following BS5950 (which is the principal code for the design of structural steelwork in the UK) ;

- Matouš *et al.* use genetic algorithms to increase the performances of composite laminated structures [MAT00] ;

- Moreau-Giraud *et al.* study in [MOR02] a coupling with a bolted rim : a torque is transmitted by adhesion using bolts placed at a certain radius. The (multicriteria) problem consists in minimizing the radius, the number of bolts and the torque ;

- Périaux *et al.* proposed a method based on genetic algorithms to optimize the shape of a nozzle which satisfies a prescribed pressure distribution on its boundary for a given flow condition [PER01] ;

- in [LAG02], two space frame structures are treated, where the breadth, the height and length of the web as well as the flange of I-shaped cross sections are the design variables, the objectives being the mass of the structure and the solution having to satisfy Eurocode 3 ;

- in [FON95a], a low-pressure spool speed governor of a gas tubine engine is optimized ;

- Jha developed an integrated computer-aided optimal design method of a plain milling cutter [JHA95] ;

- Pham developed an evolutionary method to optimize chemical engineering processes as plug-flow reactors, batch processes, etc. [PHA98] ;

- in [WIN95], many applications are collected, in fields like aerodynamic design, aircraft control, computational fluid dynamics, etc.

These selected examples illustrate the growing place that EAs have taken in engineering optimization thanks to their robustness, adaptiveness and efficiency. However, to apply EAs more particularly to pre-design optimization, the handling of multicriteria and constrained aspects has to be dealt with. This is discussed in the next chapter.

CHAPTER 3 – MULTICRITERIA OPTIMIZATION IN EVOLUTIONARY ALGORITHMS

3.1 Introduction

Though lots of optimization studies deal with only one objective, this approach is often not realistic for industrial applications. More and more real-life cases need several objectives to be handled simultaneously, for instance minimizing both the mass and the cost of a mechanical structure, – which can be a dilemma, e.g. when specially machined components are lighter but more expensive than other components, heavier but with standard pieces. Another important aspect for the designer is to obtain a product which satisfies all the constraints, i.e. all the technical requirements (related to the mechanical working of the structure, its resistance, etc.).

Therefore, after recalling some definitions essential to fathom multiobjective optimization (§ 3.2), this chapter focuses on multiobjective methods implemented in EAs (§ 3.3), and particularly multicriteria decision aid methods (§ 3.4). Then, the handling of the constraints is investigated (§ 3.5). Finally, after these bibliographical aspects, a new method is proposed, PAMUC, aiming to deal with both preferences and constraints in EAs (cf. § 3.6).

3.2 Theoretical aspects about multiobjective optimization

In industrial applications, several objectives are often pursued simultaneously (e.g. minimizing the cost and the mass of a mechanical structure, and maximizing a performance indicator at the same time). The formulation of general multiobjective (or vector) optimization problems can be written as follows :

$$\min_{x} \ \mathbf{f(x)} \tag{3.1}$$

$$s.t.: \begin{cases} \mathbf{g(x)} \geq 0, & (3.2) \\ \mathbf{h(x)} = 0, & (3.3) \\ x_i \in X_i \ \ for\ i = 1,\dots, n, & (3.4) \end{cases}$$

where :

- X_i is the set of x_i (which may be continuous, discrete or integer) ;
- $\mathbf{f(x)}^T = [\, f_1\mathbf{(x)}\ \ f_2\mathbf{(x)} \dots f_m\mathbf{(x)}\,]$ (m objectives) ;
- $\mathbf{g(x)}^T = [\, g_1\mathbf{(x)}\ \ g_2\mathbf{(x)} \dots g_p\mathbf{(x)}\,]$ (p inequality constraints) ;
- $\mathbf{h(x)}^T = [\, h_1\mathbf{(x)}\ \ h_2\mathbf{(x)} \dots h_q\mathbf{(x)}\,]$ (q equality constraints).

Here are a few definitions essential in multiobjective optimization [EHR97,THI03] :

- *Strict Pareto dominance* : a vector $\boldsymbol{u}$ is said to strictly dominate $\boldsymbol{v}$ ($\boldsymbol{u} \succ\succ \boldsymbol{v}$) if and only if $\forall i \in \{1,\ldots, m\}, u_i < v_i$ [(1)];

- *Pareto dominance* : a vector $\boldsymbol{u}$ is said to dominate $\boldsymbol{v}$ ($\boldsymbol{u} \succ \boldsymbol{v}$) if and only if $\forall i \in \{1,\ldots, m\}$, $u_i \leq v_i$ and for at least one $i \in \{1,\ldots, m\} : u_i < v_i$;

- *Weak Pareto dominance* : a vector $\boldsymbol{u}$ is said to weakly dominate $\boldsymbol{v}$ ($\boldsymbol{u} \succcurlyeq \boldsymbol{v}$) if and only if $\forall$ $\mathrm{i} \in \{1,\ldots, m\}, u_i \leq v_i$;

- *Incomparability* : two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are incomparable if neither $\boldsymbol{u} \succcurlyeq \boldsymbol{v}$ nor $\boldsymbol{v} \succcurlyeq \boldsymbol{u}$;

- *Pareto optimality* : a design vector $\boldsymbol{x}^* \in \boldsymbol{F}$ is Pareto optimal if and only if there exists no other $\boldsymbol{x} \in \boldsymbol{F}$ such that :

  $$f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*) \quad \text{for } i = 1,\ldots, m, \tag{3.5}$$

  with $f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^*)$ for at least one objective i. $\boldsymbol{F}$ is the feasible domain defined by :

  $$\boldsymbol{F} = \{ \boldsymbol{x} \in \boldsymbol{X} \mid g_j(\boldsymbol{x}) \geq 0 \;\; \text{for } j = 1,\ldots, p \text{ and } h_k(\boldsymbol{x}) = 0 \;\; \text{for } k = 1,\ldots, q \}. \tag{3.6}$$

- *Pareto set* : the set of all the nondominated solutions is called the Pareto set $\boldsymbol{P}^*$:

  $$\boldsymbol{P}^* = \{ \boldsymbol{x}^* \in \boldsymbol{F} \mid \nexists \; \boldsymbol{x} \in \boldsymbol{F} \;\; \text{such that } \boldsymbol{x} \succcurlyeq \boldsymbol{x}^* \}. \tag{3.7}$$

- *Pareto front* : the image of the Pareto set $\boldsymbol{P}^*$ in the objective function space is called the Pareto front (***PF***). Figure 3.1 represents the Pareto front for a minimization problem. It is also called the *trade-off surface*.

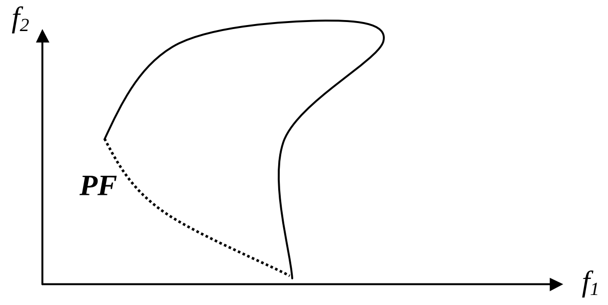


Fig. 3.1 : Pareto front PF (dotted line) in a 2-objective minimization example (figure adapted from [COE02a]).

3.3 Classification of multiobjective methods implemented in EAs

As the solution is generally not unique in vector optimization, the user has to provide additional information about his/her preferences in order to find the optimum solution. Three different approaches are available in the literature [VAN98,VAN99,ZIT00] :

[(1)] One should be very careful about the fact that these definitions of $\succ\succ$, $\succ$ and $\succcurlyeq$ are written for a *minimization* problem, which explains the "opposite" signs (e.g. $\boldsymbol{u} \succ\succ \boldsymbol{v}$ when $u_i < v_i \;\; \forall i$).

- preferences may be used at the end, when the Pareto front has been completely determined (a posteriori methods) ;
- preferences may be used during the optimization process, in an interactive way (progressive methods) ;
- preferences may be included since the beginning of the search process (a priori methods) : the user has to assign a weight to each criterion, or at least a ranking of the *m* objectives.

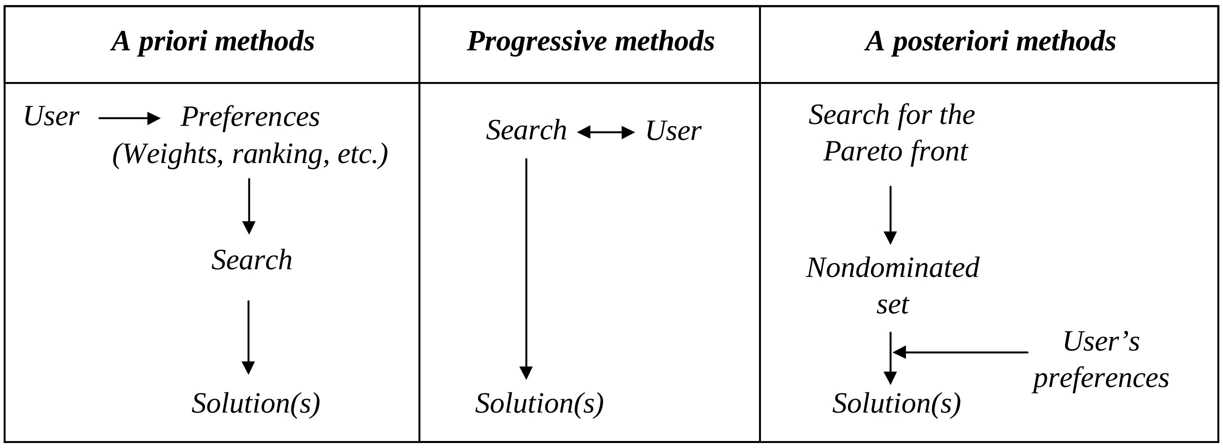


Fig. 3.2 : A priori, progressive and a posteriori methods for multiobjective optimization.

Figure 3.2 symbolically illustrates these three approaches. Multiobjective methods combined with evolutionary algorithms [ZIT99a] are briefly presented below, before focusing on a priori techniques.

It should be noted that the terms “multicriteria” and “multiobjective” are often synonymous in the literature. However, in this work, as suggested in [COE02a], the expression “multiobjective optimization” will refer solely to the presence of multiple objectives, while “multicriteria optimization” will imply the use of an additional procedure to deal with the user’s preferences.

3.3.1 A posteriori methods

In a posteriori methods, the main step consists in drawing up the shape of the Pareto front. There are various methods to find out the nondominated solutions using evolutionary algorithms (see [COE02a ,VAN00a]). As three of them (VEGA, MOGA and NSGA) play an important role in multiobjective optimization and will be mentioned again in the following of the thesis, their formulation will be briefly described below.

The first a posteriori evolutionary method, VEGA (Vector Evaluated Genetic Algorithm) was proposed by Schaffer (cited in [VAN00a]) : in this approach, the population is divided in *m* sub-populations (where *m* is the number of objective functions). Then, during the selection step, parents of each sub-population are chosen only according to the relevant objective. After that, the sub-populations are mixed together and crossover and mutation are performed on the whole population (cf. Fig. 3.3).

VEGA is a *criterion selection* technique, because fractions of population are selected upon separate objective performance [COE02a] ; but most of a posteriori methods are based on the concept of *Pareto sampling*, whose idea was first proposed by Goldberg [GOL89]. He suggested the use of nondominated ranking and selection to move a population towards the Pareto front. A variation of this approach was proposed by Srinivas and Deb [SRI94] in the famous Nondominated Sorting Genetic Algorithm (NSGA), which ranks the individuals according to nondomination, as illustrated in Fig. 3.4 for the minimization of two objectives. Each layer is composed of individuals having the same rank, hence the same fitness value, and a sharing procedure – decreasing the fitnesses of individuals which are close one to each other in the design space – is performed in order to avoid a premature convergence towards a particular region of the search space.

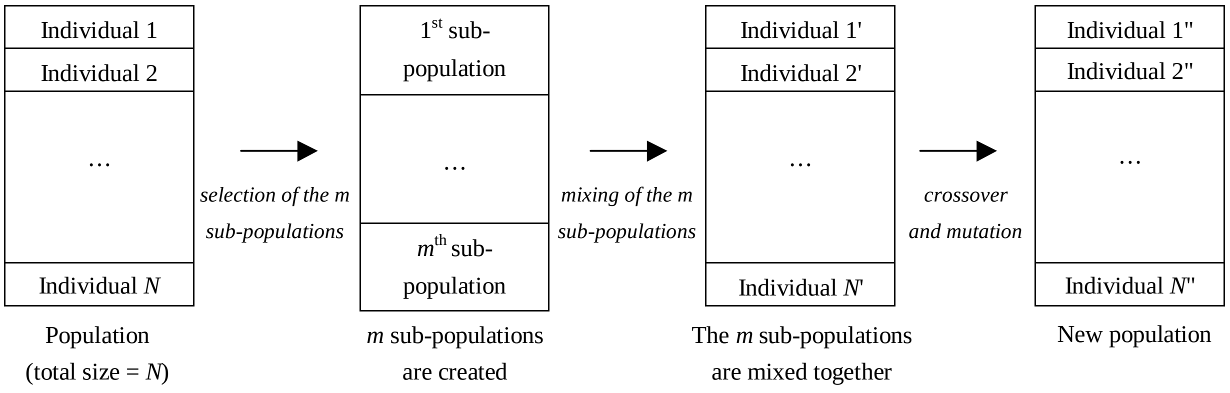


Fig. 3.3 : Description of VEGA : the j^{th} sub-population is created by selecting the best individuals following the j^{th} objective function ; then, the m sub-populations are shuffled, and crossover and mutation operators are applied to create the new generation (figure adapted from[VAN00a]).

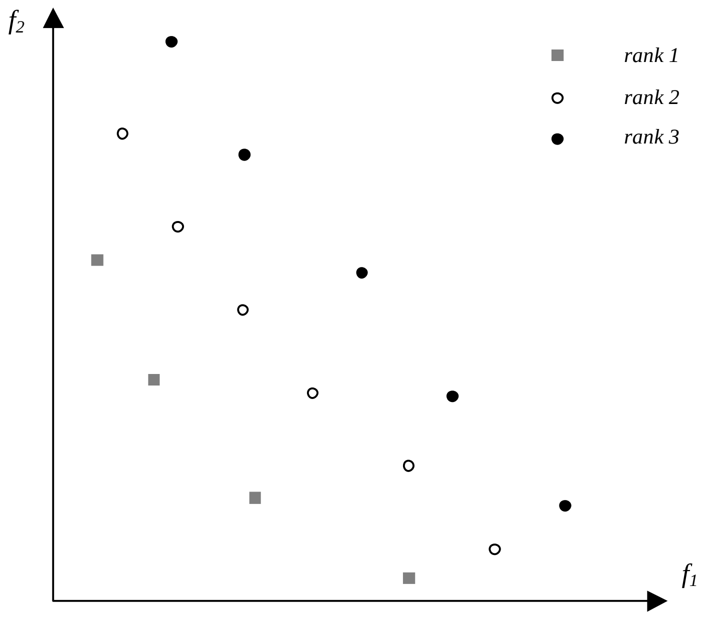


Fig. 3.4 : Description of NSGA (in a minimization problem) : ranking of the population w.r.t. nondomination (figure adapted from [SRI94]).

Another prevalent a posteriori method is the Multi-Objective Genetic Algorithm (MOGA), proposed by Fonseca and Fleming, where the rank of an individual corresponds to the number of chromosomes in the current population which dominate it [FON95,PUR01]. For instance, if an individual $\mathbf{x}_i$ (at generation t) is dominated by $p_i^{(t)}$ individuals, its rank is :

$$rank(\mathbf{x}_i, t) = 1 + p_i^{(t)}. \tag{3.8}$$

Then, the fitness function is evaluated according to the following rules :

- sort population according to rank (nondominated individuals are given rank 1) ;
- assign fitness to individuals by interpolating from the best (rank 1) to the worst (rank $n_{rank} \leq N$, where N is the size of the population) ;
- average the fitnesses of individuals with the same rank, so that all of them are sampled with the same rate.

Recent advances in a posteriori techniques include CMEA (Constraint Method-based Evolutionary Algorithm [RAN01]), NPGA2 (Niched-Pareto Genetic Algorithm 2 [ERI01]), NSGA-II (Nondominated Sorting Genetic Algorithm-II [DEB02a]), PAES (Pareto Archived Evolution Strategy [KNO00]) and SPEA2 (Strength Pareto Evolutionary Algorithm 2 [ZIT99,ZIT01]).

Once the search process is over, the user can choose a solution among the nondominated points. This generally requires a preliminary treatment of the solutions, which in some cases may be computationally expensive. For example, when lots of solutions have been found on the Pareto front, a *filtering* must be performed, to select a representative subset of the nondominated points in order to facilitate the choice for the user [OSY02] (cf. Fig. 3.5).

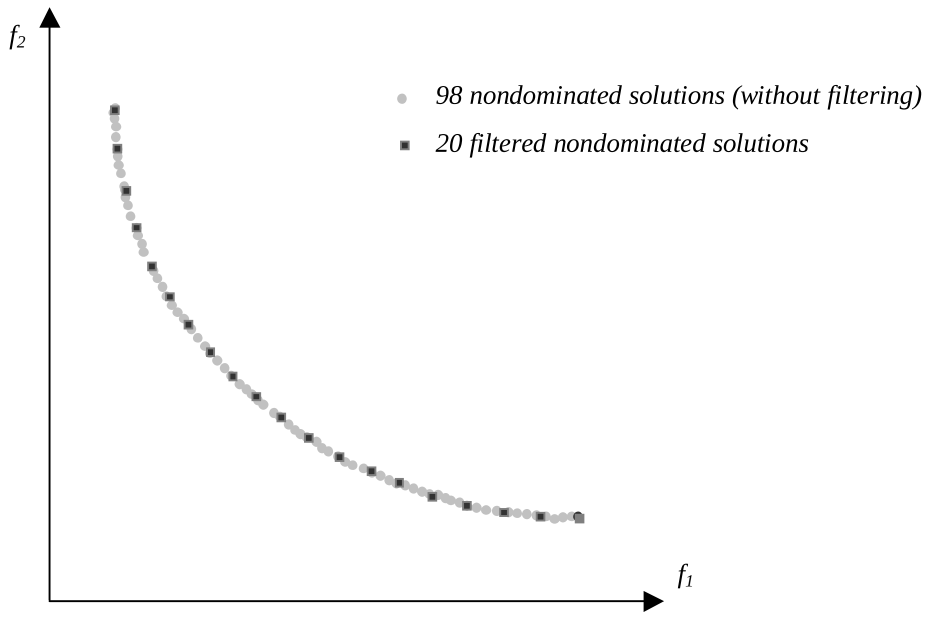


Fig. 3.5 : Process of filtering of the nondominated solutions (in order to keep a representative subset of objective vectors, 20 solutions are retained for the user) (figure adapted from [OSY02]).

3.3.2 Progressive methods

Though some methods have been developed since the 1970's to use information from the user within the search process, like the Surrogate Worth Tradeoff (SWT), or more recently Jahn's, Geoffrion's, Fandel's [COL02] or Tappeta's [TAP99] methods, Coello [COE02a] underlined that there is extremely few works dealing with interactive methods implemented in EAs, since they require an important investment of time from the decision maker.

3.3.3 A priori methods

Whereas a posteriori techniques aim to determine the shape of the whole Pareto front, and let the user decide which solution to retain, the key idea in a priori methods is to incorporate preferences since the very start of the search process, by a ranking of the objectives, or more commonly through weights assigned by the decision maker to each criterion.

This can be very useful in particular when the user already has a strong idea about his/her preferences about the objectives, or when the number of objectives exceeds three (which makes difficult and less intuitive the choice between the nondominated solutions). As a matter of fact, even with 2 or 3 objectives, making a choice after the determination of the trade-off surface is a complex task, generally requiring a preliminary treatment of the nondominated solutions, as the filtering indicated above.

Coello also noticed that there is very little work in which the preferences are explicitly handled in the evolutionary multiobjective literature [COE00b]. Therefore, in the scope of this thesis, the emphasis is put on using preferences since the beginning of the search process.

Even if it may seem redundant, it is important to dwell on the fact that there is no point in using *systematically* (i.e. for *all* multiobjective problems) either an a priori or a posteriori approach, or in deciding which approach is the better : the method selected by the user must be chosen with respect to his/her needs.

Here are the most popular a priori methods, and some of their implementations in EAs [FON95b,HOR97] :

- *lexicographic ordering* : the user has to rank the objectives following their relative importance [COE02a]. No weight is used. The optimum is thus found by minimizing the objective functions, starting with the most important one and proceeding according to the predefined order of importance. The main limitation of this approach is that when the number of objectives is high, it tends to optimize the most important ones. Furthermore, no quantitative preferences can be added to the process. This explains the low number of studies dealing with lexicographic ordering in EAs [COE02a] ;

- *weigthed sum method* : this is with no doubt the most popular a priori method in the EA community, and also among design engineers. The *m* objective functions are aggregated into one, as follows [OSY02] :

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \, f_i(\mathbf{x}), \tag{3.9}$$

  where the weights are such that :

$$\sum_{i=1}^{m} w_i = 1. \tag{3.10}$$

A thorough collection of applications of linear aggregation of objectives techniques implemented in EAs is available in [COE02a] ;

- *min-max method* : the multiobjective problem is transformed into a single-objective problem, with [OSY02] :

$$f^{*}(\boldsymbol{x}) = \max_{i\in\{1,\dots,m\}} w_i \cdot \frac{\left|f_i^0 - f_i(\boldsymbol{x})\right|}{\left|f_i^0\right|}, \tag{3.11}$$

  where f_i^0 is the separately attainable minimum of the i^{th} objective (variants of this technique adapt the formulation in order to deal with f_i^0 equal to zero), w_i are weights and f^{*} is the objective function to be minimized. In [BAL01] for instance, a min-max fitness function is used to optimize future land use and transportation plans for a city ;

- *target vector technique* : the multiobjective problem is replaced by a single-objective problem where the goal is to minimize $f^{*}(\boldsymbol{x})$ [COE02a] :

$$f^{*}(\boldsymbol{x}) = \left\| [\,\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{T}\,]\boldsymbol{W}^{-1} \right\|_{\alpha}, \tag{3.12}$$

  where $\boldsymbol{T}$ is the target vector defined by the decision maker, $\boldsymbol{W}$ a weighting matrix accounting for different scales of sizes between the m goals, and α is generally equal to 2 (Euclidean distance). The most popular variances of this technique are the *goal attainment* and the *goal programming* techniques [COL02].

  Coello presents some examples of target vector techniques incorporated in EAs in [COE02a] ; for instance, Deb uses goal programming for a welded beam design, in order to minimize both its cost and end deflection [DEB01b].

Although the methods presented above are widespread in the EA community, and generally easy to implement in standard EAs, there are also specific methods – based on the multicriteria decision aid field – to pay heed to the user’s preferences. They are discussed in the next section.

3.4 Multicriteria decision aid methods implemented in EAs

3.4.1 Introduction to multicriteria decision aid

This section presents some theoretical aspects necessary to understand the choice of the multicriteria method to implement in the EA for pre-design optimization. One of the scopes of multicriteria decision aid is to rank a set of potential solutions or candidates (called the *actions*) following several criteria $\{G_1, G_2, \dots\}$. Faced to two actions a and b, a decider will react in one of the following ways [VIN89,RUD01a] :

- $a\ I\ b$: indifference between a and b ;
- $a\ P\ b$: a is preferred to b ;
- $b\ P\ a$: b is preferred to a ;
- $a\ R\ b$: a and b are incomparable.

These three relations *{P, I, R}* are sufficient to form a *preference structure*, under the condition that *P* be asymmetric, *I* reflexive and symmetric and *R* irreflexive and symmetric [ROY93]. Every preference structure can be rigorously characterized by the definition of the relation *S* :

$$(a\ S\ b)\ \text{ iff }\ (a\ P\ b)\ \text{or}\ (a\ I\ b). \tag{3.13}$$

A *total* or *complete preorder* is defined by the fact that ∀ *a*, *b* and *c* pertaining to the set of actions *A*, Eqs. (3.14) and (3.15) hold.

$$\begin{cases} (a\ S\ b)\ \text{ or }\ (b\ S\ a) & \rightarrow \text{completeness,} \quad (3.14) \\ (a\ S\ b)\ \text{ and }\ (b\ S\ c) \Rightarrow (a\ S\ c) & \rightarrow \text{transitivity.} \quad (3.15) \end{cases}$$

In this case, all the elements of *A* can be ranked from the better to the worst, and there exists a function *G* associated to *S* such that ∀ *a, b* ∈ *A* :

$$(a\ S\ b) \Leftrightarrow G(a) \geq G(b). \tag{3.16}$$

Note that the structure defined by *S* can be modified to take into account an indifference threshold, expressing that two actions *a* and *b*, albeit different, are so close that the decider considers them as equivalent. A *threshold model* is defined in Eqs. (3.17) and (3.18) for ∀ *a, b* ∈ *A* :

$$(a\ P\ b) \Leftrightarrow G(a) \geq G(b) + Q\ , \tag{3.17}$$
$$(a\ I\ b) \Leftrightarrow |\ G(a) - G(b)| \leq Q\ . \tag{3.18}$$

with *Q* being the indifference threshold. The character relation *S* associated to the threshold model is such that ∀ *a, b, c* and *d* ∈ *A* :

$$\begin{cases} (a\ S\ b)\ \text{ or }\ (b\ S\ a), & (3.19) \\ (a\ S\ b)\ \text{ and }\ (c\ S\ d) \Rightarrow (a\ S\ d)\ \text{or}\ (c\ S\ b), & (3.20) \\ (a\ S\ b)\ \text{ and }\ (b\ S\ c) \Rightarrow (a\ S\ d)\ \text{or}\ (d\ S\ c). & (3.21) \end{cases}$$

When Eqs. (3.19) to (3.21) are satisfied, *S* defines a *quasi-order structure*.

Consequently, one of the tasks of multicriteria decision aid is to develop methods helping the decider to select a solution among a set of actions, e.g. by constructing a function *G* (like in Eqs. (3.16) or (3.17) and (3.18)) expressing his/her preferences over a set of potential solutions. To address this problem, two different approaches are encountered [ROY93] :

- the American Multi-Attribute Utility Theory (MAUT) : broadly used in the USA in economy, decision making or finance problems, the fundamental idea of this approach is to assume that every decider tries unconsciously to maximize a utility function $U = U(G_1, G_2, \ldots)$; the role of MAUT is to estimate this function [VIN89]. It is assumed that this function can be found by an iterative process, by asking judicious questions to the decider. When this function is unavailable, the task will be to identify a set of nondominated solutions. Strong preferences for one solution are established if it is clearly dominating the others. For example, Keeney and Raffia (cited in [COL02]) proposed the use of a utility function defined by Eq. (3.22) :

$$f^*(\mathbf{x}) = \prod_{i=1}^{m} [k_i\, u_i(f_i(\mathbf{x})) + 1] \tag{3.22}$$

where k_i are normalization factors ($0 \le k_i \le 1$) and u_i are strictly non decreasing functions which can incorporate nonlinearities ;

- the so-called French school, based on the outranking concept, which is built upon pairwise comparisons of the solutions. After quantifying the degree of preference (or indifference) between each pair of solutions, the rank of a solution (for a given vector of weights) is computed by comparing it to all the other solutions. In these methods, a complete pre-order structure is not always constructed : incomparability between two solutions is possible, and is part of the information given to the user to help him/her to understand the problem. The most representative instances of this approach are ELECTRE I, II and III and PROMETHEE I and II [ROY93].

As the user's information about his/her preferences has to be taken into account within the search process of EAs, it is important first to analyze the way preferences have been tackled in the evolutionary multiobjective literature.

3.4.2 Preferences in EAs

There is very little work in which the handling of preferences is explicitly dealt with in the evolutionary multiobjective literature [COE99b]. The most salient studies in this area are described below :

- in [FON93], Fonseca and Fleming combined an a posteriori method, MOGA (cf. § 3.3.1), to goal attainment (cf. § 3.3.3), in order to guide the search in a specific region of the trade-off surface. The comparison between two individuals is modified to tackle the user's preferences, expressed by means of goals. This procedure can be utilized in an a priori or an interactive way ;

- to converge towards a sub-region of the trade-off surface, Cvetković and Parmee [CVE02] used weighted Pareto optimization, which is based on the weighted dominance relation, defined as follows : for a given vector of weights $\mathbf{w}^T = [w_1 \dots w_m]$ (whose sum is equal to 1) and a real number τ (with $0 \le \tau \le 1$), $\mathbf{x}$ is said to $(\mathbf{w},\tau)$-dominate $\mathbf{y}$ iff :

$$\mathbf{x} \succcurlyeq_w^\tau \mathbf{y} \Leftrightarrow \sum_{i=1}^{m} w_i . I_{\ge}(x_i, y_i) \ge \tau, \tag{3.23}$$

where :

$$\begin{cases} I_{\ge}(x_i, y_i) = 1 & \text{if } x_i \le y_i, \quad (3.24) \\ I_{\ge}(x_i, y_i) = 0 & \text{if } x_i > y_i. \quad (3.25) \end{cases}$$

Therefore, the weighted Pareto front is the set of non dominated elements according to the $(\mathbf{w},\tau)$-dominance relation (in [CVE02], τ is set to 1). Another interesting aspect of this work is the procedure used to compute the weights. Indeed, the user has to express his/her preferences about the criteria by unary and binary relations, as in Table 3.1 :

Relation	Intended meaning
$G_1 \approx G_2$	G_1 and G_2 are equally important
$G_1 \prec G_2$	G_1 is less important than G_2
$G_1 \ll G_2$	G_1 is much less important than G_2
$\neg\, G_1$	G_1 is not important
$!\, G_1$	G_1 is important

Table 3.1 : Preference relations [CVE02].

Then, after some operations, Warshall's algorithm is applied to compute the weights [CVE00] from the user's qualitative preferences ;

- in [DEB99a], to conduct the population of the EA towards a peculiar part of the trade-off surface, a biased sharing is applied. Classical sharing procedures decrease the fitnesses of individuals which are close one to each other in the variable or the function space, in order to avoid a premature convergence towards a particular region of the search domain [BAC97]. In [DEB99a], the difference *d(i,j)* between two individuals *i* and *j* is computed as follows :

$$d(i,j) = \left(\sum_{k=1}^{m} w_k \frac{(f_k^{(i)} - f_k^{(j)})^2}{(f_k^{max} - f_k^{min})^2} \right)^{1/2} , \tag{3.26}$$

where w_k is the weight assigned to the k^{th} objective function. The biased sharing was incorporated in a NSGA (cf. § 3.3.1), and produced denser parts of the Pareto front in the region where weights guided the search ;

- Pirjanian used fuzzy rules to compute weights – whose aim is to narrow the search of the multiobjective EA – in the context of action selection of robots [PIR98] ;

- Jin and Sendhoff converted fuzzy preferences into crisped weights, or weight intervals [JIN02]. Then, either random or dynamic weighted aggregation is used [JIN01,JIN01a], both techniques initially consisting in varying the values of the weights *during* the search process, in order to cover the whole trade-off surface. These methods are adapted in [JIN02] to zoom in a specific part of the Pareto front ;

- in [SAK00], Sakawa and Yauchi adapted GENOCOP III (a constraint-handling technique for EAs : see § 3.5.1.4) to allow an interaction with the decision maker at the end of each generation of the EA. The multiobjective optimization is tackled by a min-max approach (cf. § 3.3.1) ;

- in the MAUT field, Greenwood *et al.* used elements of *imprecisely specified multi-attribute value theory* (ISMAUT) to perform imprecise ranking of attributes (cited in [COE99b]). The idea is to rank a set of solutions instead of explicitly rank the attributes (i.e. the objective functions). Preference information is also used in the survival scheme of the EA ;

- while most papers mentioned here deal with the preferences of a single decision maker, Leyva-López *et al.* are concerned about group decision, which is usually understood as the reduction of different individual preferences on a given set to a single collective preference [LEY02]. They propose an extension of the ELECTRE III multicriteria outranking methodology (cf. § 3.4.1) combined to an EA to assist a group of decision makers, in order to achieve a consensus on a set of possible alternatives ;

- Rekiek *et al.* present in [REK00,DEL01] a new method to address the hybrid assembly line design problem with several objectives. Each potential solution is a specific way of assigning a set of tasks to stations and selecting the resources to perform each of them. The multiobjective problem is solved by a grouping genetic algorithm combined with an a priori approach, PROMETHEE II (cf. § 3.4.1), wherein the user's preferences are taken into consideration by means of weights [REK01] ;

- another application of PROMETHEE II was performed by Massebeuf *et al.* who proposed an a method where PROMETHEE II is applied after the use of an a posteriori technique in order to select a subset based on the preferences of the decision maker [MAS99].

From this analysis of the state-of-the-art in the field of EAs combined with preferences, it comes out that the general trend is to identify a sub-region of the Pareto front, corresponding to a user-defined vector of weights, by using a particular multicriteria decision aid (MCDA) method. Among MCDA methods, Roy pointed out that the use of outranking ones (i.e. from the "French" school) becomes really interesting when some of the following conditions are satisfied [ROY93] :

- the criteria are heterogeneous (i.e. the functions are expressed in different scales of sizes : mass, cost, etc.) ;
- the loss on one criterion is not directly compensated by a gain in another criterion ;
- there are pseudo-criteria : for instance when the value of the j^{th} criterion is very close for two design vectors (i.e. when the relative distance between the two solutions stands below a user-defined level), one can reasonably suppose that both solutions can be considered as equivalent for that criterion ;
- the number of criteria exceeds three.

As some (or all) of these conditions are often present in the context of pre-design optimization, the outranking approach has been preferred, and more particularly PROMETHEE II. Indeed, whilst ELECTRE I, II and III and PROMETHEE I – by allowing incomparability – do not systematically furnish an aggregating function, preventing the user from having all the actions sequentially classified, PROMETHEE II is characterized by an overall function (called the net flux) which ranks all the candidates of a set, thus constructing a complete pre-order (cf. § 3.4.1). This net flux can then play the role of a fitness function and be incorporated in a standard EA for example, as it is performed in the PAMUC method proposed in this thesis

But before dissecting the core of PAMUC, another crucial matter that deserves much care in EAs is the handling of constraints. This will be discussed in the next section.

3.5 How to tackle constraints in EAs

Until now no constraints have been interfering in the discussion. However, the tackling of constraints in EAs is far from being straightforward, and circumventing this aspect is unrealistic with industrial applications, generally characterized by many constraints (e.g. physical, technical or economical requirements). After the review of the most common constraint-handling techniques in EAs for single-objective optimization (§ 3.5.1), methods specially devoted to deal with both multicriteria and constrained aspects in EAs are presented (§ 3.5.2).

3.5.1 Handling of constraints in EAs for single-objective problems

The different methods to handle constraints in EAs have been classified by Michalewicz [MIC95] and Coello [COE99a], and can be summarized as follows :

1. lethalization methods (or death penalty methods) ;
2. penalization methods ;
3. methods based on a special representation of solutions and operators ;
4. repair algorithms ;
5. methods based on a separation of objectives and constraints ;
6. hybrid methods.

They are briefly described below. Although the third and fourth kinds of methods can be very efficient, they are restricted to more specific problems (problems with linear constraints, some combinatorial applications as the traveling salesman problem, etc.). More information about these methods is available in [COE02,MIC96, REI96].

3.5.1.1 Death penalty

This is the most straightforward way to take the constraints into account : in the random creation of the initial population, and during the selection of the best individuals at each generation, the unfeasible solutions are systematically eliminated.

Van Kampen *et al.* have insisted [VAN96] on the fact that it can furnish good results in some applications, especially when the design space is convex, and when the rate of unfeasible solutions in the whole design space is not too high. Nevertheless, the main drawback of this approach is that all the unfeasible solutions share the same fitness value, so no useful information about the unfeasible domain is exploited.

3.5.1.2 Penalization methods

This is the most popular approach by the EA community. A new objective function $f^*(\mathbf{x})$ is defined by adding a penalty to each unfeasible solution. Two choices can be done while estimating a penalty function [COE02] :

- an individual might be penalized just because it is unfeasible, without taking the amount of violation into account (in this case, lethalization methods can be considered as penalty methods where the probability of selection for unfeasible solutions is equal to zero) ;
- the penalty function is computed by using the distance between the individual and the boundary of the feasible domain.

The main penalty methods are described below :

1. *Static penalty :* for each constraint, the corresponding penalty factor remains constant during all the generations [COE02] :

$$f^*(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^{k} k_j \left| g_j(\mathbf{x}) \right|^{\beta}, \tag{3.27}$$

where β is a parameter of the model (usually, $\beta = 1$ or 2). If the j^{th} constraint is satisfied ($g_j(\mathbf{x}) \geq 0$), then no penalty function must be added for this constraint ($k_j = 0$). Otherwise, k_j is a positive number which value must be defined by the user at the start of the process (and remains constant throughout the generations) ;

2. *Dynamic penalty* : static penalty factors are difficult to choose : a too large value will lead to a premature convergence close to the few feasible individuals of the initial population, while a too small value will slow down the convergence. The idea of dynamic penalties is to use low penalty factors at the earlier generations, to allow a large exploration of the search space, then to progressively increase these factors.

   For example, Joines and Houck [JOI94] proposed a technique where individuals are evaluated at each generation using the following formula :

$$f^*(\mathbf{x}) = f(\mathbf{x}) + (C.t)^{\alpha} \sum_{g_j<0} \left| g_j(\mathbf{x}) \right|^{\beta}, \tag{3.28}$$

   where t is the generation number, and C, α and β are parameters of the method (standard values proposed in [JOI94] are : $C = 1$, $\alpha = 1$ and $\beta = 2$) ;

3. *Annealing penalties* : another way to increase the importance of penalties during the process is to use penalty factors based on the principle of simulated annealing (cf. § 2.2.2). For instance, Carlson Skalak *et al.* computed the fitness of an individual by the following expression (cited in [COE02]) :

$$f^*(\mathbf{x}) = e^{-M/T}.\, f(\mathbf{x}), \tag{3.29}$$

   where M measures the amount of violation of the constraints and T (analogous to the "temperature" in cooling scheme process in simulated annealing) tends to zero as evolution progresses, such that the initial penalty factor is low and increases over time ;

4. *Adaptive penalties* : so far, no information concerning the feasibility of the solutions at earlier generations have been introduced, though it could be useful to adapt the penalty factors to the results. Smith and Tate (see [COE02]) proposed the following adaptive penalty equation :

$$f^*(\mathbf{x}) = f(\mathbf{x}) + [\, f_{adm}(t) - f_{pop}(t)] \,.\, \sum_{j=1}^{k} \left(\frac{g_j(\mathbf{x})}{q_j(t)} \right)^{c}, \tag{3.30}$$

   where :

   - $f_{adm}(t)$: value of the fitness function of the best feasible individual at generation t ;
   - $f_{pop}(t)$: value of the fitness function of the best individual at generation t ;
   - $q_j(t)$: penalties (functions of the generation t) ;
   - c : parameter of the model (usually, $c = 2$).

The goal of these techniques is to use the information about the feasible domain collected in the previous generations, in order to avoid the tedious task of setting the values of penalty coefficients (as it is the case in most static and dynamic penalty methods).

Penalization techniques are the most common way to handle constraints, because they provide good results without significant modification of the standard evolutionary algorithm. However, the difficulty in the choice of the parameters constitutes their main drawback, because no general rule can be applied to determine their values. Therefore, most recent penalization approaches use adaptive factors and parameters [COE00a,NAN01].

3.5.1.3 Decoders

In some peculiar applications, decoders offer an efficient alternative to classical penalty-based methods. Indeed, they lie on a decoding process, which can build automatically feasible solutions, thanks to instructions stored in the chromosomes [BAC97].

Though decoders have given rise to interesting results mainly in the field of combinatorial optimization (e.g. scheduling problems, pallet loading, traveling salesman problem, etc.), Kozieł *et al.* have developed a special decoder for continuous optimization [KOZ98,KOZ99], performing an homomorphous mapping between the original (often non convex) feasible domain and a "dual" convex space (defined as the hypercube $[-1,1]^n$, where n is the number of variables). Although very promising hopes have grown up with this technique, it should be pointed out that the mapping procedure can be very expensive (computationally speaking) ; furthermore, not every kind of contraint can be treated, and so far, problems with mixed variables are also excluded from this approach.

3.5.1.4 Repair strategy

Repair algorithms enjoy a huge popularity in some areas of optimization, as knapsack, set covering or traveling salesman problems. The principle is to transform an unfeasible chromosome into an admissible one, thanks to knowledge about the problem [MIC95a]. Two different implementations have been proposed :

- if $\boldsymbol{x}_u$ is an unfeasible chromosome and $\boldsymbol{x}_r$ its repaired (thus feasible) counterpart, the first approach consists in replacing the fitness function of the original individual $f(\boldsymbol{x}_u)$ by $f(\boldsymbol{x}_r)$;
- the second approach consists in replacing (with some probability) $f(\boldsymbol{x}_u)$ by $f(\boldsymbol{x}_r)$ but *also* the chromosome of $\boldsymbol{x}_u$ by the one of $\boldsymbol{x}_r$.

The first technique is related to what biologists have called the *Baldwin effect*, which assumes that the continual change and improvement of individuals in a population is due to a combination of evolution and learning. The learning (in this case the repair procedure) is transferred into the individual by means of the modified value of the fitness function [WHI94].

The second technique has similarities with *Lamarckian evolution*, which makes the hypothesis that each individual gets better during its lifetime, and this improvement is coded back into its chromosome [WHI94].

Coello [COE02] mentions the works of Liepins *et al.* for a set of combinatorial optimization problems, where repair algorithms surpass other approaches in speed and performance. Xiao *et al.* also used a repair strategy to develop an adaptive planner/navigator for mobile robots

[XIA97] : heuristic knowledge is used to move along a feasible path. Another example is provided in the field of quadratic assignment problems, where Tate and Smith create mechanisms to create automatically feasible individuals [TAT95].

Whilst most repair algorithms are concerned with combinatorial optimization, Michalewicz and Nazhiyath developed GENOCOP III for continuous optimization [MIC95b] : it is actually an hybridized algorithm, mixing coevolution (implying the presence of two distinct populations in the EA : see § 3.5.1.6) and repair strategy. The first population is composed of points satisfying only the linear constraints (which may thus be unfeasible with respect to the non linear ones), whereas only fully feasible solutions dwell in the second population.

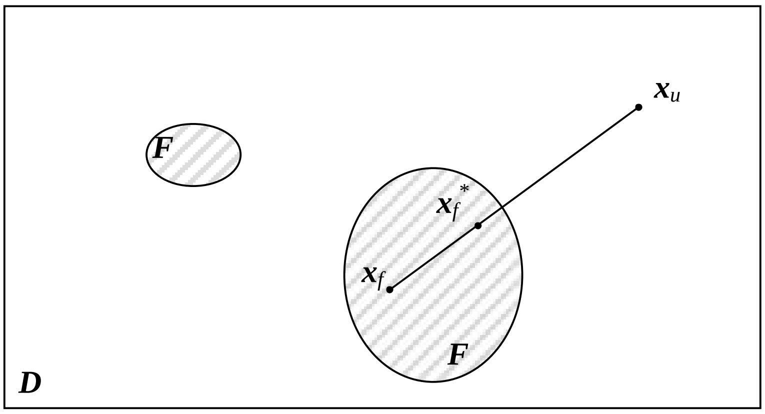


Fig. 3.6 : Description of the repairing procedure in GENOCOP III : a feasible point $\boldsymbol{x}^*_f$ is constructed from two points $\boldsymbol{x}_f$ and $\boldsymbol{x}_u$ ($\boldsymbol{F}$ is the feasible domain $\subset \boldsymbol{D}$).

Each unfeasible solution $\boldsymbol{x}_u$ is then “repaired” with some probability (only for evaluation) by selecting an admissible reference point $\boldsymbol{x}_f$ and creating a random point $\boldsymbol{x}_f^*$ located in the feasible domain, on a segment joining $\boldsymbol{x}_u$ and $\boldsymbol{x}_f$ (cf. Fig. 3.6).

One important feature of repair algorithms is the replacement probability. While a low probability may be inefficient, a too large value may lead to a premature converge. Liepins *et al.* are supporters of a *never replacing* rule (i.e. no repaired chromosome is introduced in the new population), whereas Nakano promotes an *always replacing* rule (cited in [COE02]). Besides, Orvosh and Davis (cited in [BAC97]) proposed a 5% rule in combinatorial optimization, while Michalewicz *et al.* proposed a 15% replacement rule for continuous problems. As it can be seen by these various recommendations, no systematic rule can be adopted: the replacement probability is a supplementary parameter, eminently problem-dependent, to be tuned by the user.

3.5.1.5 Constraint-preserving operators

For some specific constraints, genetic operators of the EA (like crossover and mutation) can be tailored to preserve the feasibility of the population [BAC97]. For example, if equality and inequality constraints are linear, and if each variable set is connex, then uniform, boundary and nonuniform mutations and arithmetical crossover automatically transform feasible parent(s) into offspring automatically satisfying the constraints. Like the decoders and the repair algorithms, this technique is applicable only in restricted applications.

3.5.1.6 Methods making a distinction between objective(s) and constraints

In [COE02], Coello classified the methods separating the objective(s) and the constraints in four approaches :

- *coevolutionary algorithms* : as already mentioned above (see GENOCOP III in § 3.5.1.4), coevolutionary models use two populations evolving in parallel. Paredis (cited in [COE02]) proposed a rather different version of coevolution : the first population contains the constraints to be satisfied (constraints highly violated have also a high fitness value), and the second one is composed of potential solutions (individuals with high fitness values represent solutions which respect a lot of constraints). The general idea is to focus the search on constraints that are harder to satisfy [CRA01] ;

- *superiority of feasible points* : the key idea is to put the emphasis on the feasibility (compared to the value of the objective function), specially during the selection scheme, as in Powell and Skolnick's penalty method (PS), whose penalized objective function $f^*(\mathbf{x})$ is first computed as follows :

$$f^*(\mathbf{x}) = f(\mathbf{x}) + R \,.\, \sum_{\substack{j=1 \\ g_j<0}}^{p} \left| g_j(\mathbf{x}) \right|^{\beta} \,, \tag{3.31}$$

  where R must be tuned by the user (and generally $\beta = 1$; cited in [DEB00]). Then, the value of $f^*(\mathbf{x})$ for unfeasible individuals is raised by an amount λ computed to make the fitness of the best unfeasible solution equal to the fitness of the worst feasible solution. In [DEB00], Deb proposed a constraint-handling method based on a tournament selection operator (TS), which works as follows :

  - any feasible solution is preferred to any unfeasible solution ;
  - among two feasible solutions, the one having a better objective function value is preferred ;
  - among two unfeasible solutions, the one having smaller constraint violation is preferred.

  The corresponding fitness function devised by Deb is (with β=1) :

$$f^{fitness}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } g_j(\mathbf{x}) \geq 0 \;\; \forall j = 1,\dots, p, \quad (3.32) \\[2ex] f_{max} + \sum\limits_{j=1}^{p} \left| g_j(\mathbf{x}) \right|^{\beta} & \text{otherwise.} \quad (3.33) \end{cases}$$

  These two techniques are very close, except that Deb's method does not require the tuning of the additional parameter R. Another technique, CONGA (COnstraint based Numeric Genetic Algorithm), was proposed by Hinterding and Michalewicz [HIN98] : in the first part of the process, the algorithm only looks for feasible solutions, without taking the objective function into consideration. Then, as the amount of feasible individuals increases, the algorithm focuses on improving the best ones. To perform this operation, two selection schemes are used : the first one choosing an individual either for mutation or as a parent (called $\mathbf{x}_1$) for crossover, following the same criteria as Deb's (see above) ; the second one choosing a mate for $\boldsymbol{x}_1$ by selecting the individual with the least number of satisfied constraints in common with $\boldsymbol{x}_1$. The idea is to cross complementary individuals in order to produce better ones ;

- *behavioural memory* : Schoenauer and Xanthakis proposed a method where the constraints are treated sequentially [SCH93]. Indeed, their algorithm works as follows :

  1. set $j = 1$ (j being the number of the current treated constraint) ;
  2. evolve the population to minimize the violation of the j^{th} constraint ;
  3. set $j = j + 1$;
  4. evolve the current population to minimize the violation of the j^{th} constraint. During this step, the solutions which do not satisfy the $j - 1$ previous constraints are eliminated (cf. death penalty in § 3.5.1.1) ;
  5. go back to step 3 as long as $j < p$.

  Though this method requires that constraints could be ordered, which is as often as not unrealistic, it can be well suited for some specific problems (as the generation of software test data : see [SCH93]) ;

- *multiobjective optimization techniques* : in this approach, multiobjective techniques are used to handle the constraints. The first method was proposed by Parmee and Purchase (cited in [MEZ02]), and consisted in using VEGA (cf. § 3.3.1) by considering the constraints as objectives, to reach a feasible region of the search space.

  Another implementation of VEGA was done by Surry *et al.*, in the so-called COMOGA (Constrained Optimization by Multi-Objective Genetic Algorithms) method, which functions as follows [SUR95] :

  1. compute the constraint violations for all solutions ;
  2. perform a Pareto ranking based on constraint violation (e.g. by counting the number of individuals of the population dominated by each solution) ;
  3. compute the value of the “true” objective function (e.g. the cost of the pipes in the gas network problem studied in [SUR95]) for each member of the population ;
  4. select a proportion p_{cost} of individuals based on the “true” objective function, and the others on constraint ranking ;
  5. apply the recombination operators (crossover, mutation) ;
  6. adjust p_{cost} to make the rate of feasible individuals become closer to the user-defined rate of feasible solutions τ. Lowering p_{cost} favours feasible solutions, while raising p_{cost} improves the “true” objective function (e.g. decreases the cost).

  A third use of VEGA was performed by Coello in [COE00c], where at each generation, the population is split into $p + 1$ sub-populations (with p equal to the number of constraints), where the “true” objective function and the p constraints play the role of the $1 + p$ objective functions. The first sub-population is related to the objective function, and the $j + 1^{\text{th}}$ sub-population uses the following fitness function :

  - if $g_j(\boldsymbol{x}) < 0$ then fitness $= g_j(\boldsymbol{x})$;

- elseif $v \neq 0$ then fitness = $-v$;
- else fitness = $f(\mathbf{x})$,

where $g_j(\mathbf{x})$ is the j^{th} constraint, v refers to the number of violated constraints and f is the objective function.

In [CAM97], Camponogara *et al.* proposed to transform the single-objective problem into a 2-objective one, with the "true" objective function and a function $\Phi(\mathbf{x})$ (to be minimized) related to the constraints :

$$\Phi(\mathbf{x}) = \sum_{j=1}^{p} max\ [0, g_j(\mathbf{x})]. \tag{3.34}$$

The multiobjective method used in [CAM97] consists in separating the current population in layers of nondominated solutions (as in NSGA : see Fig. 3.4) ; then, from pairs of points $\{x_i , x_j\}$ located on different layers, a line search algorithm is applied in the direction joining both points (see Fig. 3.7).

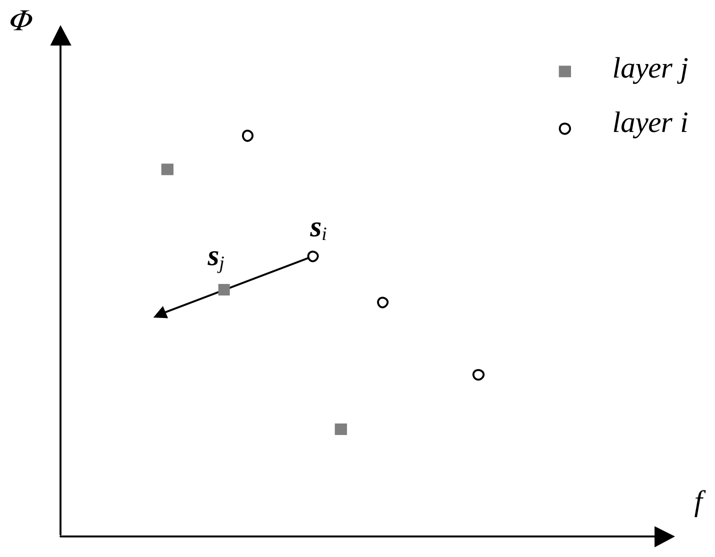


Fig. 3.7 : Example of line search direction obtained from points $\mathbf{s}_i$ and $\mathbf{s}_j$ in the method proposed in [CAM97] ; f represents the "true" objective function while Φ is related to the satisfaction of the constraints.

In [JIM99], Jiménez and Verdegay used a selection scheme close to the one proposed by Deb in [DEB00] (see Fig. 3.7), where :

- any feasible solution is preferred to any unfeasible solution ;
- among two feasible solutions, the one having better objective function value is preferred ;
- among two unfeasible solutions, the one having the lowest maximum violation constraint wins.

The main difference with [DEB00] is that no additional procedure is used in [JIM99] to preserve the diversity of the population.

Ray *et al.* (cited in [COE02]) developed an approach in which solutions are ranked separately following the value of their objective and constraints ; then, mating restrictions (using specific information about the feasibility of each member of the population) are applied.

In [RUN00], Runarsson and Yao, noting that it is difficult to find a proper value for the penalty parameters in penalization methods, proposed the use of a stochastic ranking : a probability P_f of using only the objective function for comparisons while ranking unfeasible individuals is introduced. It means that given any pair of individuals, the probability of comparing them following the objective function is equal to 1 if both are feasible, and to P_f otherwise. This is close to the idea proposed by Surry *et al.* (see above ; cf. [SUR95]), except that here P_f is not self-adaptive, and remains constant during the whole optimization process.

In [OSY02], Osyzcka proposes to apply the constraint tournament selection method – initially devoted to multiobjective problems – to single-objective constrained optimization. Basically, it is based on a set of selection rules near to the ones proposed in [JIM99] and [DEB00] :

- any feasible solution is preferred to any unfeasible solution ;
- among two unfeasible solutions, the one having the lowest maximum violation constraint wins ;
- among two feasible solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$: if $\boldsymbol{x}_1$ dominates $\boldsymbol{x}_2$ (resp. $\boldsymbol{x}_2$ dominates $\boldsymbol{x}_1$), $\boldsymbol{x}_1$ (resp. $\boldsymbol{x}_2$) is selected ; otherwise an individual is randomly chosen.

Finally, two other extensions of multiobjective a posteriori techniques were proposed to handle constrained problems : in [AGU03], Aguirre *et al.* introduced the IS-PAES (Inverted-Shrinkable Pareto Archived Evolutionary Strategy), which is an extension of the PAES developed by Knowles and Corne [KNO00], and in [COE02c], Coello and Mezura implemented a version of NPGA (Niched-Pareto Genetic Algorithm) adapted to tackle the constraints.

3.5.1.7 *Hybrid methods*

The last category of constraint-handling methods combines evolutionary techniques with other approaches, as Lagrange multipliers (as in gradient-based algorithms), ant colonies, fuzzy logic, etc. They are summarized in [COE02].

3.5.1.8 *Remarks*

In this bibliographical overview of constraint-handling techniques in EAs, emphasis was put on two approaches : first, the penalization methods, since they are the most widely used in the EA realm – and specially in structural optimization –, and secondly the methods based on multiobjective optimization algorithms.

More and more techniques attempt to prune away user-defined parameters, by using self-adaptivity, which exempts the user from tuning specific coefficients, as in penalization techniques for example. Furthermore, it is well known in the EA community that incorporating specific knowledge about the domain, albeit very efficient, is applicable only in restricted problems

[COE02]. Therefore, to deal with mechanical design optimization, which can have very different mathematical formulations, a robust tool is needed to handle the constraints.

The techniques mentioned above were mainly applied in single-objective optimization. Indeed, in [COE02a], Coello insisted on the point that only a few papers encompass both multicriteria and constrained features in EAs. This twofold aspect will be discussed in the next section.

3.5.2 Handling of constraints in EAs for multiobjective problems

The first way to handle constraints in multicriteria EAs was to add a penalty to the objective functions of the unfeasible individuals, as in single-objective optimization (cf. Richardson *et al.* [RIC89], Kundu [KUN99]). The other popular approach consists in considering the constraints as additional objectives [COE02a] ; multiobjective EAs applied to constrained single-optimization have been presented in § 3.5.1.6, and are easily extended to multiobjective optimization. Besides, Kurpati *et al.* suggested to add specific improvements to some a posteriori multiobjective methods [KUR02].

So far, few papers deal with both constrained and preferences aspects in EAs, and it has been emphasized in [COE02a] that there is a lack of development of constraint-handling techniques especially devoted to multiobjective EAs. A novel method is thus presented below to tackle both aspects.

3.6 Preferences Applied to MUltiobjectivity and Constraints (PAMUC)

3.6.1 Motivation

The bibliographic study presented above shows the overwhelming number of methods created to handle either multiobjective or constrained optimization with EAs. However, when the simultaneous tackling of both aspects is considered, very few specific methods are available. Furthermore, while the amount of a posteriori methods is high, mainly because it does not require any additional information about the user's preferences, less techniques integrate those preferences within the search process.

As mentioned in § 3.4.2, the use of outranking methods (from the field of multicriteria decision aid) is recommended by Roy [ROY93] when at least one of the following conditions is satisfied :

- the criteria are heterogeneous ;
- the loss on one criterion is not directly compensated by a gain in another criterion ;
- there are pseudo-criteria ;
- the number of criteria exceeds three.

These conditions are often encountered in design optimization. Thus, to take the constraints into account, the satisfaction of the constraints will be considered as a new objective (as in many multiobjective approaches). From the pro and contra of the several methods cited above, it seems obvious that the main features required to build a reliable multicriteria and constraint-handling design optimization method are the following :

- to heed the constraints, the introduction of additional parameters, whose tuning is time-consuming and problem-dependent, should be avoided ;
- as a very small difference for a criterion between two potential solutions does not imply that one can be preferred to the other, indifference thresholds must be taken into account ;
- to be applicable in a wide range of problems, it should be implemented without major modification of the standard EA.

Consequently, the idea is to use an aggregating technique (i.e. where the user assigns a weight to each objective following its relative importance), namely PROMETHEE II, to deal with multicriteria optimization [FIL02a]. Incorporating PROMETHEE II in an EA has already been performed by Rekiek ([REK01] ; see § 3.4.2), but in the particular case of an assemby line design problem.

The goal is thus to extend this approach to any optimization problem (by integrating PROMETHEE II to a *standard* EA). Moreover, a specific technique is also required to tackle the constraints. Therefore, the simultaneous handling of both (multicriteria and constrained) features led to the development of a novel method : PAMUC (Preferences Applied to MUltiobjectivity and Constraints).

The main characteristics of PROMETHEE II are explained in the next section. Then, PAMUC is described in § 3.6.3.

3.6.2 Description of PROMETHEE II

PROMETHEE II (Preference Ranking Organisation METHod for Enrichment Evaluations – 2nd version) is due to Brans and Mareschal [BRA86]. It is characterized by a particular scaling of the objective functions, and by the fact that each individual a is compared to all the other solutions of the set E in order to build the net flux (or preference flow), which measures the quality of a compared to the rest of E [BRA96].

Here are the outlines of PROMETHEE II :

1. For each objective f_i , a preference function $P_i(a,b)$ is created, which allows to compare any couple (a,b) of individuals (cf. Fig. 3.8). As in [REK01], a linear preference function with indifference was implemented. This choice is discussed below.

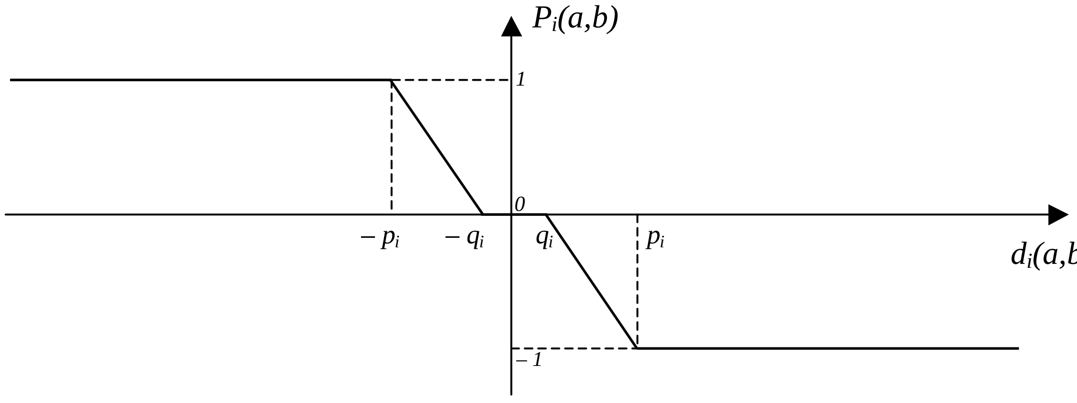


Fig. 3.8 : Linear preference function with indifference threshold (for a minimization problem) in PROMETHEE II (figure adapted from [BRA86]).

The parameters p_i (preference thresholds) and q_i (indifference thresholds) must be defined by the user for each objective. If $f_i(a) \neq 0$, $d_i(a,b)$ is computed as follows :

$$d_i(a,b) = \frac{f_i(a) - f_i(b)}{|f_i(a)|} \quad \text{if } f_i(a) \neq 0. \tag{3.35}$$

Otherwise, $P_i(a,b)$ is computed as follows :

$$\begin{cases} \text{if } f_i(a) > f_i(b) : P_i(a,b) = -1, & (3.36) \\ \text{if } f_i(a) = f_i(b) : P_i(a,b) = 0, & (3.37) \\ \text{if } f_i(a) < f_i(b) : P_i(a,b) = 1. & (3.38) \end{cases}$$

2. Then, the preference index of *a* over *b* is defined by :

$$\pi(a,b) = \sum_{i=1}^{m} w_i \,.\, P_i\,(a,b) \qquad \left(\text{with } \sum_{i=1}^{m} w_i = 1\right). \tag{3.39}$$

   The weights w_i reflect the relative importance assigned to each objective, and are chosen by the user ;

3. Finally, to compare a solution *a* with the *N–1* other solutions of a set *E*, the preference flow $\phi(a)$ is computed as follows :

$$\phi(a) = \frac{1}{N-1} \,.\, \sum_{\substack{b \in E \\ b \neq a}} \pi(a,b). \tag{3.40}$$

Consequently, the multiobjective problem is transformed into the maximization of the preference flow ϕ, which acts like the fitness function of a single-objective problem.

The theoretical framework introduced in § 3.4.1 is useful now to shed light on some characteristics of PROMETHEE II. This method performs a quasi-order structure, which means that :

- *indifference* is taken into account, thanks to the definition for each criterion of the indifference thresholds q_i, whose value must be determined by the user following his/her knowledge about the objectives ;

- compared to the other outranking methods (like ELECTRE I, II and III or PROMETHEE I), PROMETHEE II excludes *incomparability*. Incomparability appears for instance when two individuals are mutually nondominated (one with respect to the other). Though it may bring helpful information about the multiobjective problem, it is a delicate task to integrate this concept into an aggregating method ;

- a direct consequence of the previous point is that for each pair *(a,b)* of individuals of the EA population, either *a S b* or *b S a* (where *S* is the relation defined in § 3.4.1). At each generation, the population of the EA can thus be ranked upon a *single criterion* (the preference flow) aggregating all the objectives, and playing the role of the fitness function.

Then, to take constrained and multicriteria aspects simultaneously into account, a novel approach is proposed in the next section.

3.6.3 Description of PAMUC

The aim of this original method called PAMUC (*Preferences Applied to MUltiobjectivity and Constraints*) is to solve constrained multicriteria problems (with preferences defined by the user) [FIL02b]. As in most multiobjective evolutionary methods, the constraints are considered as a new objective ; it consists thus in using PROMETHEE II with *m+1* objectives : the *m* objective functions and one more related to the satisfaction of the constraints (the equality constraints are transformed in inequalities as in Eq. (2.5) mentioned above). This latter function related to the satisfaction of the constraints is formulated as follows :

$$f^{(m+1)} = \sum_{g_j<0} \frac{|g_j(x)|}{k_j}. \tag{3.41}$$

The factors k_j are scaling factors which can be estimated thanks to this formula :

$$k_j = \frac{\sum_{n=1}^{N} \left| g_j^{(t)}(x) \right|}{N} , \tag{3.42}$$

- N : population size ;
- $g_j^{(t)}(\boldsymbol{x})$: value of the j^{th} constraint for $\boldsymbol{x}$ at generation t.

The flow-chart of the algorithm is illustrated in Fig. 3.9.

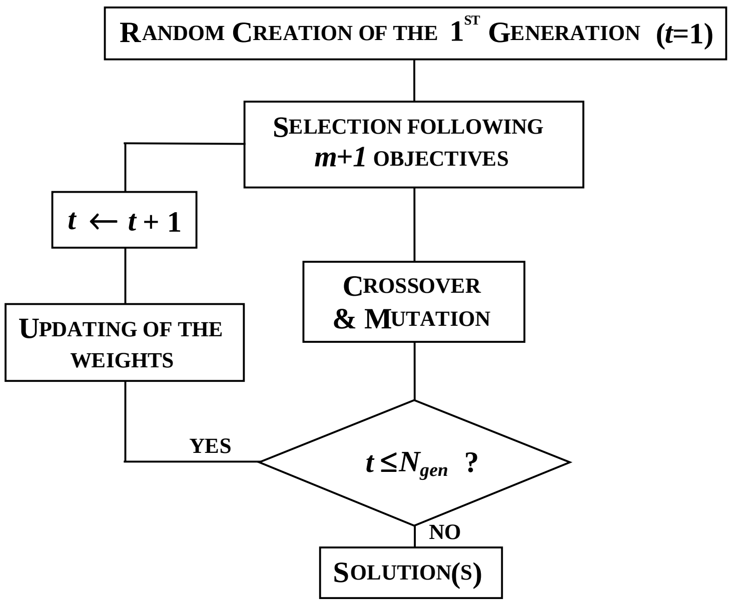


Fig. 3.9 : Flow-chart of PAMUC.

The difference with a standard single-objective unconstrained EA lies mainly in the selection. Before the selection scheme, the population at generation t is evaluated by using PROMETHEE II with $m+1$ objectives. The selection procedure is a binary tournament operator, which functions as follows : when two new individuals (the children) are created by crossover and mutation, they are compared with their parents thanks to the PROMETHEE II method. Then, the two best individuals among the four (i.e. amid the parents and their children) are selected to take part of the next generation.

As in any a priori method, weights (w_i^*) are initially chosen by the user for the m objectives, but in order to take into account the objective related to the satisfaction of the constraints, the actual weights used in PROMETHEE II are computed in the following way :

$$w_i^{(t)} = w_i^{*}.RF^{(t)} \qquad \text{for } i = 1,\dots, m, \tag{3.43}$$

$$w_{m+1}^{(t)} = 1 - RF^{(t)}. \tag{3.44}$$

- $RF^{(t)}$ is the ratio of the population which satisfies all the constraints at the previous generation ;
- at the first generation : $w_{m+1}^{(1)} = 1$ and $w_i^{(1)} = 0$ for $i = 1,\dots, m$.

One can easily check that :

$$\sum_{i=1}^{m+1} w_i^{(t)} = 1. \tag{3.45}$$

The weights are adaptive (as the coefficients in adaptive penalty-based methods : see § 3.5.1.2) : when the number of feasible individuals is low (which is generally the case at the first generations), the relative importance given to the $m+1^{th}$ objective (satisfaction of the constraints) is high. Then, if a growing part of the population tends to satisfy the constraints, a decrease of $w_{m+1}^{(t)}$ automatically occurs.

3.6.4 Choice of the weights

Though the decision maker usually chooses the weights intuitively, some approaches have been developed to help the user in reflecting appropriately his/her preferences.

In [MAR89], Mareschal mentioned a family of methods where the user is asked to supply constraints about the weights. For example, in a 5-objective problem, if the user wants to express that the first three criteria must not exceed in importance the last two ones, Eq. (3.46) can be written :

$$w_1 + w_2 + w_3 \geq w_4 + w_5 . \tag{3.46}$$

Each time a new constraint is added, the space of potential weights progressively narrows until proper values are obtained. Another approach consists in using qualitative information about the user's preferences, like in [CVE00] (see § 3.4.2), or in [EZZ00] with PROMETHEE I and II. A graphical interpretation of the weights have also been developed by Brans *et al.* in the PROMCALC-GAIA software : more information can be found in [BRA94].

Nevertheless, in this work, no effort is done to incorporate an additional method to compute weights : it is assumed that the decision maker can determine their values intuitively.

3.7 Conclusions

This chapter was first devoted to expose the way multicriteria and constrained aspects are taken into account in EAs. The review of the litterature showed that although lots of techniques have been developed to deal with either multicriteria or constrained aspect, few methods explicitly deal with both features. Therefore, a novel method was proposed : PAMUC. Its goal is to solve multiobjective constrained problems, where the user incorporates his/her preferences about the objectives since the very start of the search process, by means of weights. It consists in considering the satisfaction of the constraints as a new objective, and using PROMETHEE II, a multicriteria decision aid method, to rank the members of the EA population at each generation. Besides, adaptivity of the weights is applied in order to converge more easily towards the feasible domain.

The validation strategy, numerical results and discussion about the advantages and drawbacks of PAMUC are presented in the next chapter.

CHAPTER 4 – VALIDATION OF THE PAMUC METHOD

4.1 Introduction

The PAMUC method designed to solve multicriteria constrained optimization problems was described in the previous chapter. In order to validate it, two aspects have to be considered : its ability to reach the feasible domain – hence to find an optimal solution satisfying all the constraints –, and its efficiency in reflecting correctly the user's preferences.

The global strategy of validation will be presented in § 4.2, whence it will be concluded that whilst comparing two solutions is straightforward in single-objective optimization, this task is much more critical with multiobjective problems. Therefore, lots of metrics have been proposed to express the quality of one method compared to another ; the choice of the most appropriate metric to validate PAMUC will be discussed in § 4.3. Then, numerical results will be presented (§ 4.4), followed by general remarks and conclusions about PAMUC (§ 4.5).

4.2 Strategy of validation

4.2.1 Implementations of PAMUC

In order to validate PAMUC, the standard evolutionary algorithm (Std-EA), implemented in Matlab (cf. § 2.3.2), was applied on standard test cases [FIL03].

Then, PAMUC was implemented in C++ in the framework of the existing EAs of Boss Quattro (Samtech s.a.), a commercial software for optimization and parametrical studies [BOS01], which contains a real-coded and a binary-coded EA. It has been used for the parametrical optimization of four valves designed by Techspace Aero (Snecma group) for launcher Ariane 5 (this is the scope of Chapter 6).

It must be underlined that since EAs are stochastic algorithms (random numbers are used at the creation of the initial population and in the operations of selection, crossover and mutation), they have to be launched several times for each problem in order to make statistics about the results.

Furthermore, all examples were applied with $p_i = 1$ and $q_i = 0$. These parameters define respectively the (relative) preference and indifference thresholds defined by the user (for each objective) when 2 individuals are compared (cf. § 3.6.2). Those values will be discussed in § 4.4.3.

4.2.2 Single-objective constrained optimization (SOCO) problems

As shown above, PAMUC can handle single-objective as well as multiobjective problems without major modification. Consequently, the efficiency of PAMUC in tackling the constraints is first presented in § 4.4.1 on a set of 8 challenging single-objective constrained benchmarks :

- 3 examples from the classical test case suite from Hock and Schittkowski [HOC81], solved by PAMUC and Joines and Houck's penalty-based technique (implemented in the Std-EA with parameters $C = 1$, $\alpha = 1$ and $\beta = 2$ as proposed in [JOI94]) :

    - S-HED : a heat exchanger design problem ;
    - S-3EQ : a problem characterized by 3 equality constraints ;
    - S-6ACT : an example where 6 constraints are active at the optimum ;

- 4 single-objective constrained test cases proposed by Deb [DEB00] :

    - S-CRES : an example with a crescent-shape feasible domain ;
    - S-38IC : a problem with a high number of inequality constraints (38) ;
    - S-0.5F : a problem where only 0.5% of the admissible space is feasible ;
    - S-HIM : Himmelblau's problem ;

  In the first three test cases (S-CRES, S-38IC and S-0.5F), results of PAMUC are compared with solutions obtained in [DEB00] by two methods (cf. § 3.5.1.6) :

  1. Powell and Skolnick's penalty method (PS), whose penalized objective function is given by Eq. (3.31) ;
  2. Deb's constraint-handling method based on a tournament selection operator (TS), which was described in § 3.5.1.6.

  In the fourth example (S-HIM), results of PAMUC are compared with solutions from different studies collected by Coello in [COE02] ;

- 1 engineering application (S-WBD) : a welded beam design, cited in [COE02] and tested with PAMUC and Joines and Houck's penalty-based technique.

4.2.3 Multiobjective optimization (MOO) problems

After investigating the constrained aspect in single-objective examples, multiobjective problems have been tested.

In design optimization, the most popular a priori approach is the classical weighted sum method [COE02a], because of its simplicity of implementation and its intuitiveness. Therefore, PAMUC is compared to the weighted sum method (§ 4.4.2) where Joines and Houck's penalty-based technique is used to tackle the constraints (with parameters $C = 1$, $\alpha = 1$ and $\beta = 2$).

If two different points $\mathbf{z}_a$ and $\mathbf{z}_b$ obtained by two a priori multiobjective methods are compared (i.e. for a given value of the set of weights), three situations can occur : (1) $\mathbf{z}_a$ dominates $\mathbf{z}_b$, (2) $\mathbf{z}_a$ and $\mathbf{z}_b$ are nondominated or (3) $\mathbf{z}_b$ dominates $\mathbf{z}_a$ (cf. Fig. 4.1).

Here, as no specific combination of weights is preferred in the validation, the following procedure is applied for 2-objective optimization problems (cf. § 4.3.5 for an extension to m-objective problems) : the EA is run n_{runs} times – with either the weighted sum method or PAMUC –, each time with different values for the weights (varying from $\{w_1^* = 1 ; w_2^* = 0\}$ to $\{w_1^* = 0 ; w_2^* = 1\}$ by a constant step). Then, from the sets of solutions found by both methods, the dominated ones are discarded, and finally the Pareto sets are compared.

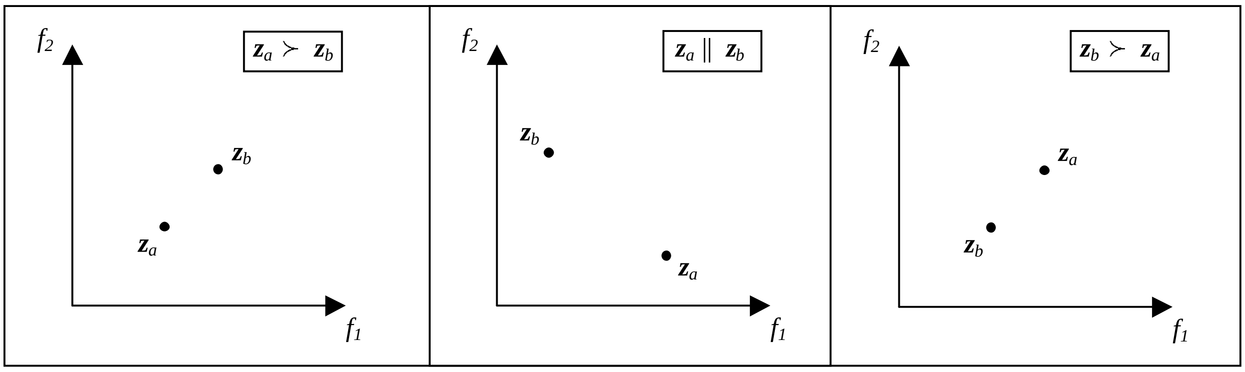
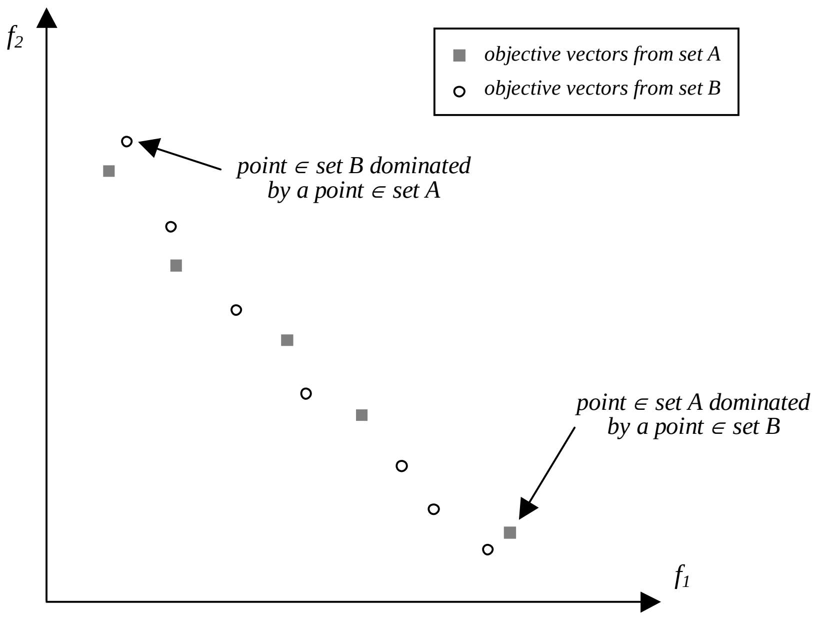


Fig. 4.1 : Comparison of two points $\mathbf{z}_a$ and $\mathbf{z}_b$ (in a two-objective minimization example) : (1) $\mathbf{z}_a$ dominates $\mathbf{z}_b$, (2) $\mathbf{z}_a$ and $\mathbf{z}_b$ are incomparable and (3) $\mathbf{z}_b$ dominates $\mathbf{z}_a$ (figure adapted from [ZIT03]).

In a general way, 2 nondominated sets *A* and *B* found by two different multiobjective methods are often interwoven, i.e. some solutions from the set *A* can dominate solutions from the set *B*, and vice versa, making the comparison not straightforward (cf. Fig. 4.2). Therefore, specific metrics have been developed [ZIT02], and are discussed in the next section.



f_2
■ objective vectors from set A
○ objective vectors from set B
point ∈ set B dominated by a point ∈ set A
point ∈ set A dominated by a point ∈ set B
f_1

Fig. 4.2 : Comparison of two nondominated sets A and B in a 2-objective minimization problem : the sets A and B are incomparable (figure adapted from [ZIT03]).

The 9 multiobjective benchmarks treated by the weighted sum method and PAMUC are :

- 1 unconstrained example proposed by Cvetković [CVE00] : M-UC (with a convex Pareto front PF) ;
- 2 examples collected in [VAN98], and Osyczka :
  - M-LOC : due to Kita, it has linear objectives and constraints (convex PF) ;
  - M-LOQC : due to Osyczka, it has linear objectives and quadratic constraints (concave PF) ;

- 3 examples taken in [DEB01] :
    - M-QOC (quadratic objectives and constraints) ;
    - M-DPF (discontinuous PF) ;
    - M-LFS (feasible space composed of layers) ;

- 1 engineering application : BDP (the beam design problem defined in [OSY02]) ;

- 2 three-objective problems due to Viennet (collected in [COE02a]) :
    - M-3OU (unconstrained test case) ;
    - M-3OC (constrained test case).

The numerical results are presented in § 4.4.2.

4.3 Validation of multiobjective optimization methods

4.3.1 Introduction to metrics for comparing nondominated sets

Before describing the main metrics (or quality indicators) used in multiobjective optimization, some theoretical notions are required. The definitions introduced in § 3.2 to compare two objective vectors can be extended to the comparison of two nondominated sets [ZIT03] :

- a set A *strictly dominates* a set B ($A \succ\succ B$) iff every objective vector $\boldsymbol{z}^B \in B$ is strictly dominated by at least one point $\boldsymbol{z}^A \in A$;

- a set A *dominates* a set B ($A \succ B$) iff every objective vector $\boldsymbol{z}^B \in B$ is dominated by at least one point $\boldsymbol{z}^A \in A$;

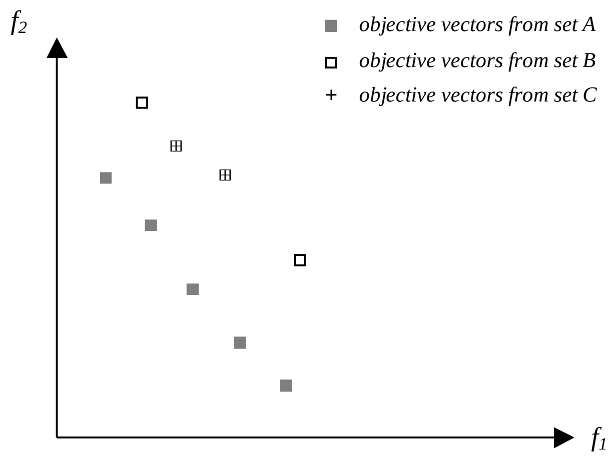
- a set A is *better* than a set B ($A \rhd B$) iff every objective vector $\boldsymbol{z}^B \in B$ is weakly dominated by at least one point $\boldsymbol{z}^A \in A$ and $A \neq B$;

- a set A *weakly dominates* a set B ($A \succcurlyeq B$) iff every objective vector $\boldsymbol{z}^B \in B$ is weakly dominated by at least one point $\boldsymbol{z}^A \in A$;

- 2 sets A and B are incomparable ($A \parallel B$) iff neither A weakly dominates B nor B weakly dominates A.

These definitions are illustrated on three nondominated sets A, B and C in Fig. 4.3, and will be used below in the comparison of the quality indicators (cf. § 4.3.4).

Two kinds of measures have been developed in the multiobjective literature : unary metrics and binary metrics [ZIT03]. Both categories will be described respectively in §§ 4.3.2 and 4.3.3, and the choice of the best suited metric to compare PAMUC to the weighted sum method will be discussed in § 4.3.4.



Relations between A, B and C :

$A \succ\succ B \qquad A \succ\succ C$

$A \succ B \qquad A \succ C$

$A \triangleright B \qquad A \triangleright C \qquad B \triangleright C$

$A \succeq B \qquad A \succeq C \qquad B \succeq C$

$A \succeq A \qquad B \succeq B \qquad C \succeq C$

Fig. 4.3 : Comparison of three nondominated sets A, B and C following the definitions introduced in § 4.3.1 (in a 2-objective minimization problem) (figure adapted from [ZIT03]).

4.3.2 Unary measures

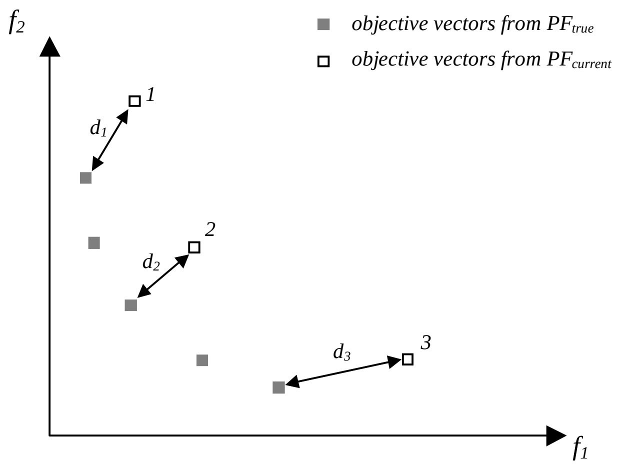
Unary quality indicators are very attractive since they can be used independently, i.e. they give a value which does not relie on the objective vectors of another set [ZIT03]. Van Veldhuizen *et al.* [VAN00] and Knowles [KNO02a] made a thorough review of all the unary metrics used in multiobjective evolutionary optimization. The most representative metrics can be classified in two families :

- the measures of the *closeness to the Pareto front* : these metrics attempt to measure how close to the true Pareto front (denoted PF_{true}) a nondominated set found by an a posteriori method (such as VEGA, MOGA, NSGA, etc.) is. They need the user to know the theoretical Pareto front, and are often used (for validation of a new method) during the EA process to check the convergence of the algorithm. For example, the error ratio (cf. [COE02a]) reports at each generation of the EA the finite number of vectors which are members of PF_{true}. Another metric mentioned in [VAN00b] is the generational distance GD, representing how far in average the current set $PF_{current}$ is from PF_{true} :

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}, \tag{4.1}$$

  where n is the number of vectors in $PF_{current}$ and d_i is the Euclidean distance between each objective vector of $PF_{current}$ and the closest element of PF_{true}, as depicted in Fig. 4.4. Variants of this metric are the maximum Pareto front error and the average Pareto front error (cf. [COE02a]).

- the measures of the *distribution* of the objective vectors along the trade-off surface. In the hyperarea metric (or S-metric) proposed by Zitzler and Thiele [ZIT02] for example, the measure is defined by the area covered by the objective vectors of $PF_{current}$ (cf. Fig. 4.5). Schott proposed another metric for 2-objective problems – the spacing S –, computing explicitely the spread of the objective vectors throughout $PF_{current}$:



f_2
f_1
objective vectors from PF_{true}
objective vectors from $PF_{current}$
1
2
3
d_1
d_2
d_3

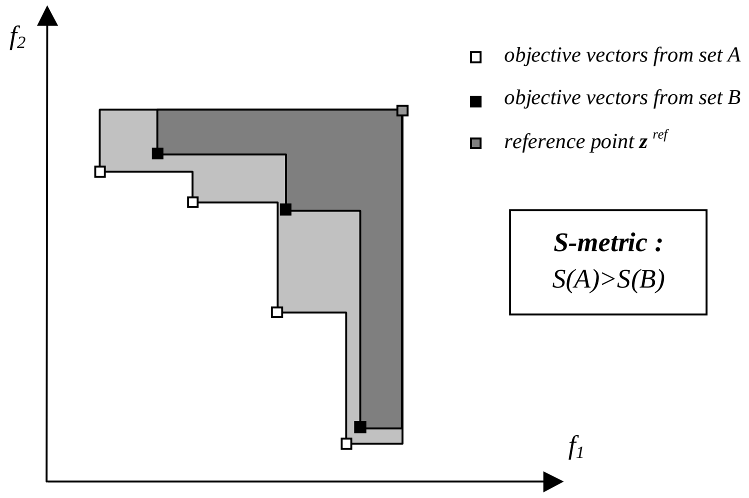
Fig. 4.4 : Set of nondominated objective vectors ($PF_{current}$) compared to PF_{true} thanks to the generational distance GD (in a 2-objective combinatorial minimization problem) (figure adapted from [COE02a]).

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\overline{d} - d_i\right)^2}\ , \tag{4.2}$$

with :

$$d_i = \min_j \left(\left|\mathbf{z}_1^i - \mathbf{z}_1^j\right| + \left|\mathbf{z}_2^i - \mathbf{z}_2^j\right|\right),\ i, j = 1,\ldots, n, \tag{4.3}$$

and where n is the number of vectors in $PF_{current}$, $\mathbf{z}_\alpha^i$ is the α^{th} component of $\mathbf{z}^i$ ($\in PF_{current}$) and $\overline{d}$ is the mean of all d_i .



f_2
f_1
objective vectors from set A
objective vectors from set B
reference point $\mathbf{z}^{ref}$
S-metric :
S(A)>S(B)

Fig. 4.5 : Computation of the S-metric (in a 2-objective minimization problem) : from a reference point $\mathbf{z}^{ref}$ (chosen by the user), the rectangles encompassed between $\mathbf{z}^{ref}$ and each vector of the set A (resp. B) define a shaded surface whose area is the S-metric of A (resp. B) (figure adapted from [KNO02a]).

Other unary quality indicators can be found in [COE02a] and [KNO02a].

4.3.3 Binary measures

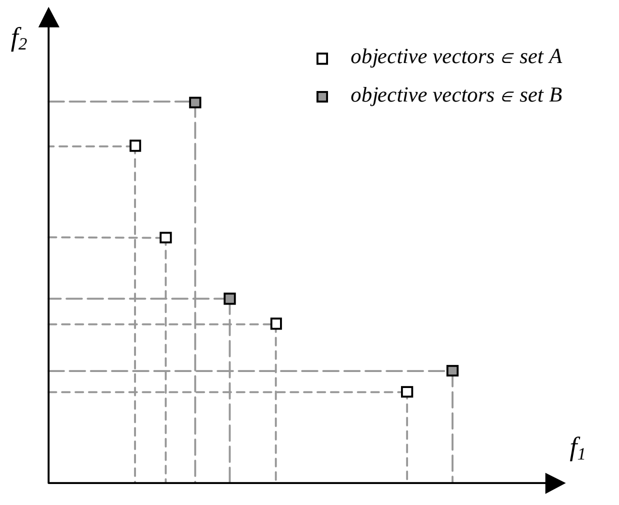
While an important amount of unary indicators are available in the literature, fewer metrics are specially devoted to compare pairs of sets. The three main binary metrics are described below (the other binary metrics are generally mere extensions of unary metrics : see [ZIT03]) :

- the *C-metric* proposed by Zitzler and Thiele [ZIT00], whose aim is to compute the relative coverage of two nondominated sets *A* and *B*. The measure is given by a pair of indicators *C(A,B)* and *C(B,A)*, calculated as follows :

$$C(A,B) = \frac{\left|\left\{ \mathbf{z}^B \in B \,\middle|\, \exists\, \mathbf{z}^A \in A : \mathbf{z}^A \succcurlyeq \mathbf{z}^B \right\}\right|}{|B|}, \tag{4.4}$$

$$C(B,A) = \frac{\left|\left\{ \mathbf{z}^A \in A \,\middle|\, \exists\, \mathbf{z}^B \in B : \mathbf{z}^B \succcurlyeq \mathbf{z}^A \right\}\right|}{|A|}, \tag{4.5}$$

  where $|A|$ is the cardinal of *A* and $|B|$ is the cardinal of *B*. *C(A,B)* = 1 means that all decision vectors in *B* are weakly dominated by at least one point $\in$ *A*, whilst *C(A,B)* = 0 means that none of the points in *B* is weakly dominated by a point in *A*. Figure 4.6 illustrates the *C*-metric on a 2-objective example.



C-metric :

- *|A| = 4 and |B| = 3*
- *Number of points of A dominated by at least one point of B = 0*
- *Number of points of B dominated by at least one point of A = 2*

$\rightarrow$ *C(A,B) = 2/3 ≈ 0.666*
$\rightarrow$ *C(B,A) = 0/4 = 0*

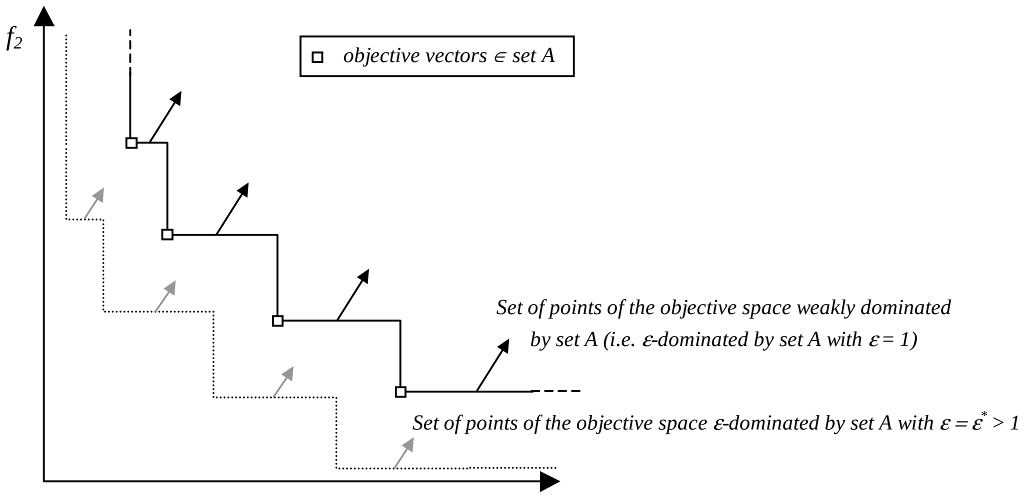
Fig. 4.6 : Illustration of the C-metric on a 2-objective minimization example (figure adapted from [ZIT00]).

- in [ZIT03], Zitzler and Thiele suggested the use of a *binary ε-indicator*, based on the definition of the ε-dominance : for a given $\varepsilon > 0$, a vector $\mathbf{z}^A$ is said to ε-dominate a vector $\mathbf{z}^B$ ($\mathbf{z}^A \succcurlyeq_\varepsilon \mathbf{z}^B$) iff :

$$\forall 1 \le i \le m : z_i^A \le \varepsilon \,.\, z_i^B, \tag{4.6}$$

  where *m* is the number of objective functions (cf. Fig. 4.7). Then, the indicator I_ε is defined by :

$$I_\varepsilon(A,B) = \inf_{\varepsilon \in \mathbb{R}} \{ \forall \mathbf{z}^B \in B, \exists \mathbf{z}^A \in A : \mathbf{z}^A \succeq_\varepsilon \mathbf{z}^B \}. \quad (4.7)$$



f_2
□ objective vectors ∈ set A
Set of points of the objective space weakly dominated by set A (i.e. ε-dominated by set A with ε = 1)
Set of points of the objective space ε-dominated by set A with ε = ε* > 1

Fig. 4.7 : Illustration of the ε-dominance (figure adapted from [ZIT03]).

For example, if $I_\varepsilon(A,B) \leq 1$ and $I_\varepsilon(B,A) > 1$, then the set A is *better* than B ($A \rhd B$).

- in [HAN98], Hansen and Jaszkiewicz proposed the *R1*-norm, based on the computation of the probability that a set A is better than another set B over a family of utility functions U. It is formally defined by Eq. (4.8) :

$$R1(A,B,U,p) = \int_{u \in U} C(A,B,u)\, p(u)\, du, \quad (4.8)$$

where :

$$C(A,B,u) = \begin{cases} 1 & \text{if } u^*(A) > u^*(B), \quad (4.9) \\ ½ & \text{if } u^*(A) = u^*(B), \quad (4.10) \\ 0 & \text{if } u^*(A) < u^*(B), \quad (4.11) \end{cases}$$

with A and B are the sets to compare, U is a set of utility functions $u : \mathbb{R}^m \rightarrow \mathbb{R}$ which maps each point from the objective space into a measure of utility, and $u^*(A)$ is defined as follows :

$$u^*(A) = \max_{\mathbf{z} \in A} \{ u(\mathbf{z}) \}, \quad (4.12)$$

and similarly for $u^*(B)$. Finally, $p(u)$ is the probability density of $u \in U$. A value of $R1(A,B,U,p)$ close to 1 would show that A outperforms B, whereas a value near 0 would show that B is better than A. It can be easily demonstrated that :

$$R1(A,B,U,p) = 1 - R1(B,A,U,p). \quad (4.13)$$

Another metric presented in [HAN98] is the *R2*-norm, which is built directly from the value of u^*, without using the outcome function $C(A,B,u)$:

$$R2(A,B,U,p) = \int_{u \in U} (u^*(A) - u^*(B))p(u)du. \quad (4.14)$$

A variant of this metric, the *R3*-norm, uses the ratio between $u^*(A)$ and $u^*(B)$ instead of the difference.

4.3.4 Discussion

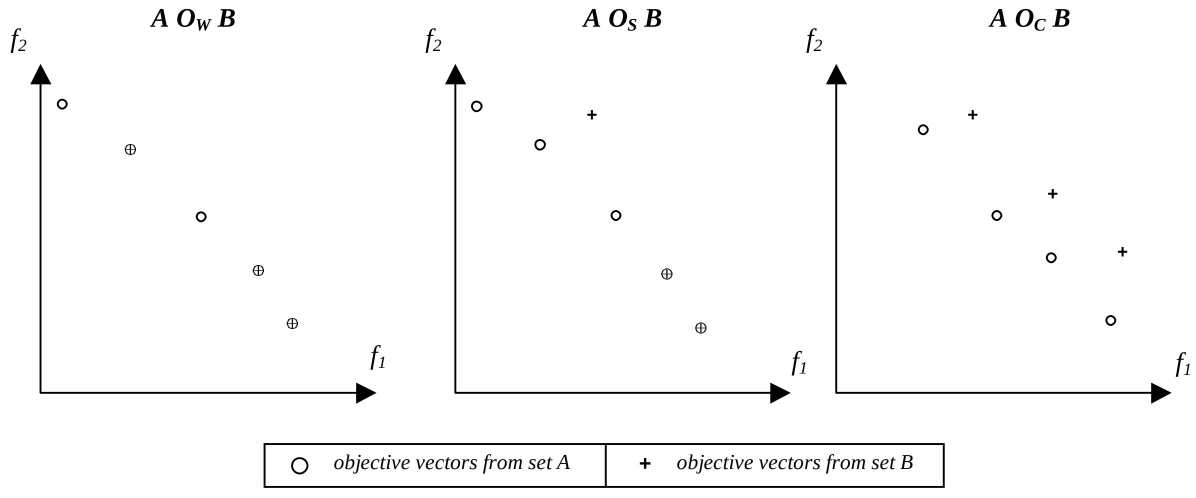
To address the problem of comparing those different metrics, the new concept of *outperformance* [HAN98] has to be introduced in addition to the definitions given in § 4.3.1 :

- a set *A weakly outperforms* a set *B* ($A\ O_W\ B$) iff $A \neq B$ and for each objective vector $\mathbf{z}^B \in B$, $\mathbf{z}^B$ is weakly dominated by at least one point $\mathbf{z}^A \in A$;
- a set *A strongly outperforms* a set *B* ($A\ O_S\ B$) iff for each objective vector $\mathbf{z}^B \in B$, $\mathbf{z}^B$ is weakly dominated by at least one point $\mathbf{z}^A \in A$, and $\exists\ \mathbf{z}^{B*} \in B$ such that $\mathbf{z}^{B*}$ is dominated by a point in *A* ;
- a set *A completely outperforms* a set *B* ($A\ O_C\ B$) iff for each objective vector $\mathbf{z}^B \in B$, $\mathbf{z}^B$ is dominated by at least one point $\mathbf{z}^A \in A$.

This is illustrated in Fig. 4.8. A careful examination of these definitions shows that :

$$A\ O_W\ B \Leftrightarrow A \rhd B \text{ (A is better than B)}, \quad (4.15)$$

$$A\ O_C\ B \Leftrightarrow A \succ B \text{ (A dominates B)}. \quad (4.16)$$



A O_W B
A O_S B
A O_C B
f2
f1
○ objective vectors from set A
+ objective vectors from set B

Fig. 4.8 : Illustrations of the weak, strong and complete outperformance definitions in 2-objective minimization examples (figure adapted from [HAN98]).

Finally, a last couple of definitions based on these relations must be added to the collection of concepts needed to analyze rigorously the advantages and drawbacks of the different metrics [HAN98] :

- a metric M is *weakly compatible* with the outperformance relation O (where O stands for O_W, O_S or O_C) iff for each pair of sets A and B such that $A\ O\ B$, $M(A,B)$ will evaluate A as being not worse than B ;

- a metric M is *compatible* with the outperformance relation O iff for each pair of sets A and B such that $A\ O\ B$, $M(A,B)$ will evaluate A as being superior to B.

An in-depth theoretical study made by Zitzler and Thiele [ZIT03] shows that unary metrics are in general not capable of indicating whether a set is superior to another one ; it allows at best to infer that a set is not worse than another one. Moreover, in some cases, if a set A is evaluated superior to a set B, it can be that B actually overcomes A, even when different unary metrics are used and give the same trend.

For example, Knowles *et al.* indicate in [KNO02] that results obtained with the S-metric are strongly dependent from the reference point, as depicted in Fig. 4.9. From a more practical point of view, unary metrics often require the knowledge of the theoretical Pareto front PF_{true}, which is seldom available, typically in industrial applications. Consequently, whereas unary metrics are very useful to quantify the closeness to the Pareto front or the spread of the nondominated solutions along the trade-off surface, using them to compare nondominated sets must be done with great care.

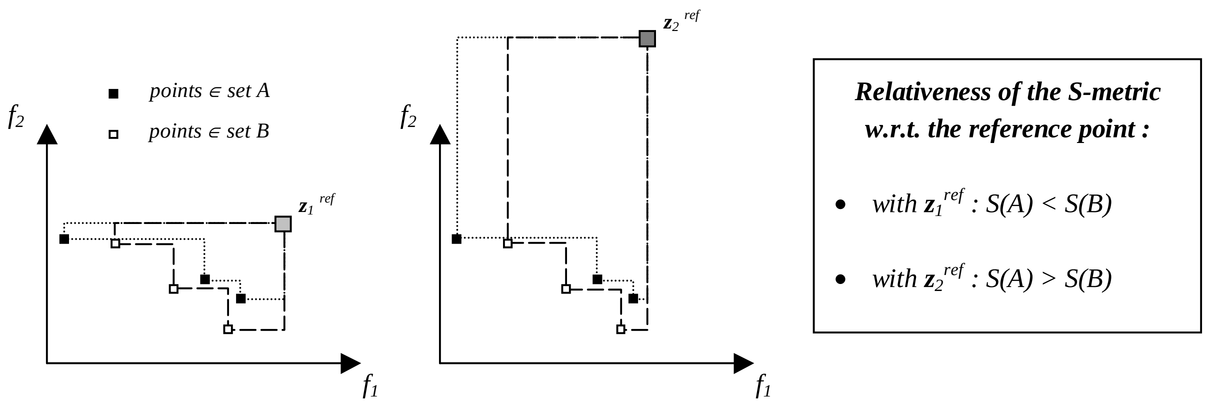

■ points ∈ set A
□ points ∈ set B
f_2
f_1
$\mathbf{z}_1^{ref}$
f_2
f_1
$\mathbf{z}_2^{ref}$
Relativeness of the S-metric w.r.t. the reference point :
with $\mathbf{z}_1^{ref}$: $S(A) < S(B)$
with $\mathbf{z}_2^{ref}$: $S(A) > S(B)$

Fig. 4.9 : 2-objective example illustrating the relativeness of the S-metric with respect to the reference point (figure adapted from [KNO02a]).

Therefore a binary metric seems more suited to compare PAMUC to the weighted sum method. First, the C-metric is investigated. Knowles reported in [KNO02a] some of its potential drawbacks :

- in some configurations, the C-norm can induce a cycling of the relations between 3 nondominated sets A, B and C, i.e. a situation where A is superior to B, B is superior to C and C is superior to A ;

- if the sets A and B to compare are of different cardinality and/or the distribution of the points along the set is non-uniform, the results may be unreliable ;

- it is not compatible with the weak outperformance relation O_W.

The second binary measure introduced in § 4.3.3 is the binary ε-indicator. Zitzler and Thiele note in [ZIT03] that although it can detect if a set A is *better* than a set B ($A \rhd B$), meaning that it is compatible with the outperformance relation O_W, it is not compatible with the complete outperformance relation O_C. Moreover, it involves much computational overhead.

Finally, the last metrics investigated are Hansen and Jaszkiewic's norms. It has been shown in [KNO02a] that they are in general compatible with the three outperfomance relations O_W, O_S and O_C (the compatibility depending on the set of utility functions U used in the computation of the norms) ; therefore their use is recommended by Knowles and Corne [KNO02]. *R1*-norm has been preferred to *R2*- and *R3*-norms because it does not need to make the hypothesis that it is meaningful to add the value of different utility functions of U.

The following section dissects the *R1*-nom applied in this work to assess PAMUC in comparison with the weighted sum method.

4.3.5 Focus on Hansen & Jaszkiewicz's *R1*-norm

The general procedure of computing *R1*-norm is described in [HAN98], and it will be developed in detail in this section, including the specific options taken in this work. The purpose is to compare two sets called S^{PAMUC} and S^{WS} (which are nondominated objective vectors obtained respectively with PAMUC and the weighted sum method).

The first step is to define the set of utility functions U. A utility function $u : \mathbb{R}^m \rightarrow \mathbb{R}$ is a preference model which maps each point in the objective space into a value of utility (which the decision maker wants to be maximized). The most convenient way to build U is to use parameterized utility functions $u(\mathbf{z},\mathbf{r})$ where $\mathbf{r} \in D(\mathbf{r})$ and $D(\mathbf{r})$ is the variation domain of $\mathbf{r}$.

When no preference of the decision maker is incorporated (which is the case since S^{PAMUC} and S^{WS} are obtained by varying uniformly the weights, without giving emphasis to one part of the trade-off surface with regard to another one), Jaszkiewicz [JAS01] suggests the use of weighted Tchebycheff utility functions $u_\infty(\mathbf{z},\mathbf{z}^*,\Lambda)$ (where λ_j are the weights, for $j = 1,\ldots, m$) :

$$u_\infty(\mathbf{z}, \mathbf{z}^*, \Lambda) = -\max_j \{ \lambda_j.(z_j^* - z_j)\}, \tag{4.17}$$

where $\mathbf{z}^*$ is the ideal point with respect to S^{PAMUC} and S^{WS} (cf. § 3.2) :

$$z_j^* = min \{ z_j \mid \mathbf{z} \in S^{PAMUC} \cup S^{WS}\}. \tag{4.18}$$

As no particular set of weights is preferred, U is the set defined by all combinations of $u_\infty(\mathbf{z},\mathbf{z}^*,\Lambda)$ with $\Lambda = \{\lambda_1,\ldots, \lambda_m\}$ such that each component λ_j varies from 0 to 1 by a constant step, and of course :

$$\sum_{i=1}^{m} \lambda_i = 1. \tag{4.19}$$

For example, Tables 4.1 and 4.2 indicate all combinations of weights defining $u_\infty(\mathbf{z},\mathbf{z}^*,\Lambda)$ in U in a 2-objective (with a step equal to 0.25) and a 3-objective problem (with a step equal to ⅓) .

λ_1	λ_2
0	1
0.25	0.75
0.50	0.50
0.75	0.25
1	0

Table 4.1 : Combinations of weights for a 2-objective problem with a constant step = ¼.

λ_1	λ_2	λ_3
0	0	1
0	⅓	⅔
0	⅔	⅓
0	1	0
⅓	0	⅔
⅓	⅓	⅓
⅓	⅔	0
⅔	0	⅓
⅔	⅓	0
1	0	0

Table 4.2 : Combinations of weights for a 3-objective problem with a constant step = ⅓.

Then, $u^*(S^{PAMUC})$ and $u^*(S^{WS})$ are defined as follows :

$$u^*(S^{PAMUC}) = \max_{\mathbf{z}\in S^{PAMUC}} \left\{ u_\infty(\mathbf{z},\mathbf{z}^*,\Lambda) \middle| \mathbf{z} \in \mathbf{S}^{PAMUC} \right\}, \tag{4.20}$$

$$u^*(S^{WS}) = \max_{\mathbf{z}\in S^{WS}} \left\{ u_\infty(\mathbf{z},\mathbf{z}^*,\Lambda) \middle| \mathbf{z} \in \mathbf{S}^{WS} \right\}. \tag{4.21}$$

The construction of u^* for two sets A and B is illustrated in Fig. 4.11 on a simple 2-objective example with 3 points in each set : A = {(2,6) ; (4,4) ; (8,2)} and B = {(2,8) ; (4,6) ; (7,1)} (cf. Fig. 4.10) and $\Lambda = \{\lambda_1 , \lambda_2\}$ where λ_1 (resp. λ_2) varies linearly from 0 (resp. 1) to 1 (resp. 0) as expressed in Eq. (4.22) :

$$\Lambda = \{\lambda_1 , \lambda_2\} = \{ t , 1 - t \} ; t \in [0,1]. \tag{4.22}$$

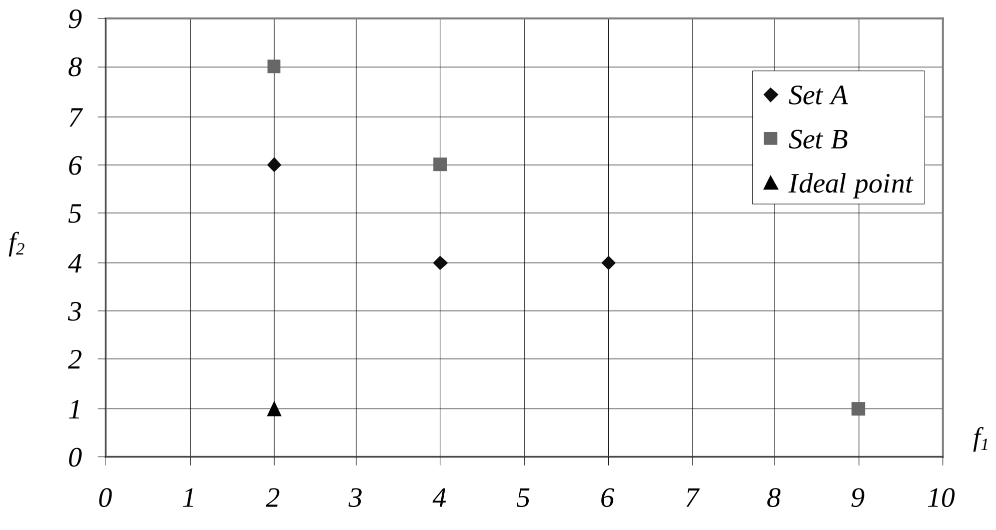


Fig. 4.10 : Definition of sets A and B in the objective space.

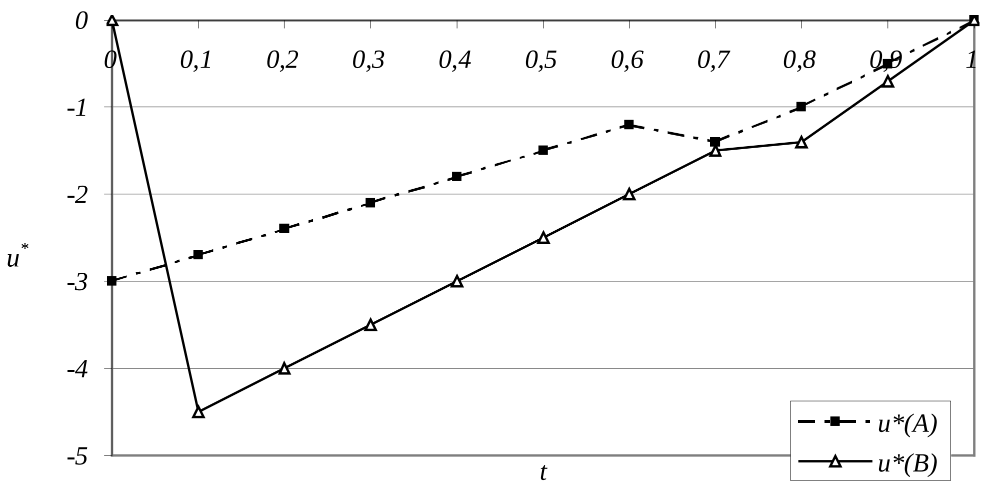


Fig. 4.11 : Function u^* for 2 sets A and B with respect to the parameter t defining Tchebycheff weights ($u^*(A)$ is superior to $u^*(B)$ for every value of t from t = 0.1).

Then, an outcome function is introduced in order to compare S^{PAMUC} and S^{WS} :

$$C(S^{PAMUC}, S^{WS}, u) = \begin{cases} 1 & \text{if } u^*(S^{PAMUC}) > u^*(S^{WS}) \,, & (4.23) \\ ½ & \text{if } u^*(S^{PAMUC}) = u^*(S^{WS}) \,, & (4.24) \\ 0 & \text{if } u^*(S^{PAMUC}) < u^*(S^{WS}) \,. & (4.25) \end{cases}$$

Finally, the *R1* measure can be built to reflect the probability that S^{PAMUC} is better than S^{WS}, by integrating over all the utility functions :

$$R1(S^{PAMUC}, S^{WS}, U, p) = \int_{u \in U} C(S^{PAMUC}, S^{WS}, u)\, p(u)\, du, \qquad (4.26)$$

where *p(u)* is the probability of *u* to be chosen by the user. In the validation process, each set of weights has the same probability of being chosen by the user, thus *p(u)* has a uniform distribution.

The meaning of *R1*-norm is the following : a value of $R1(S^{PAMUC}, S^{WS}, U, p)$ close to 1 would show that S^{PAMUC} is superior to S^{WS}, whereas a value near 0 would lead to an opposite conclusion. It can be easily demonstrated that *R1*-norm is *symmetric* (in the sense defined in [KNO02]), i.e.:

$$R1(S^{PAMUC}, S^{WS}, U, p) = 1 - R1(S^{WS}, S^{PAMUC}, U, p). \qquad (4.27)$$

Now that the methodology of validation of PAMUC has been fully explained, it is due time for presenting the numerical results.

4.4 Numerical results

4.4.1 Single-objective constrained optimization (SOCO)

4.4.1.1 Test case S-HED

The first single-objective constrained example is a heat exchanger design problem (test case 106 in [HOC01]), characterized by 8 variables and 6 inequality constraints :

$$\min f(\mathbf{x}) = x_1 + x_2 + x_3 \qquad (4.28)$$

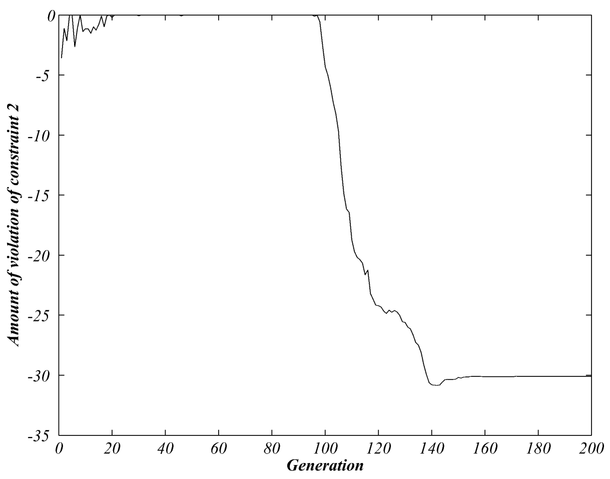
$$\begin{aligned}
\text{subject to : } & g_1(\mathbf{x}) = 1 - 0.0025\,(x_4 + x_6) \geq 0, & (4.29)\\
& g_2(\mathbf{x}) = 1 - 0.0025\,(x_5 + x_7 - x_4) \geq 0, & (4.30)\\
& g_3(\mathbf{x}) = 1 - 0.01\,(x_8 - x_5) \geq 0, & (4.31)\\
& g_4(\mathbf{x}) = x_1 x_6 - 833.33252\, x_4 - 100\, x_1 + 83333.333 \geq 0, & (4.32)\\
& g_5(\mathbf{x}) = x_2 x_7 - 1250\, x_5 - x_2 x_4 + 1250\, x_4 \geq 0, & (4.33)\\
& g_6(\mathbf{x}) = x_3 x_8 - x_3 x_5 + 2500\, x_5 - 1250.10^{3} \geq 0, & (4.34)\\
& 100 \leq x_1 \leq 10000, & (4.35)\\
& 1000 \leq x_2 \leq 10000, & (4.36)\\
& 1000 \leq x_3 \leq 10000, & (4.37)\\
& 10 \leq x_i \leq 1000,\ i = 4,\ldots, 8. & (4.38)
\end{aligned}$$

The EA parameters used to solve this problem are gathered in Table 4.3.

Symbol	*Parameter*	*Value*
Env	*Environment*	*Std-EA Matlab*
Cod	*Coding of the variables*	*Real*
N	*Size of the population*	*50*
N_{gen}	*Number of generations*	*200*
T_s	*Type of selection*	*Tournament*
n_t	*Number of individuals participating to a tournament*	*2*
p_c	*Probability of crossover*	*1*
T_c	*Type of crossover*	*SBX*
p_m	*Probability of mutation*	*Variable**
T_m	*Type of mutation*	*PBM*
η_c	*Distribution index for crossover*	*1*
η_m	*Distribution index for mutation*	*Variable**

Table 4.3 : EA parameters for test case S-HED (cf. § 2.3.2.6).*

The problem was tackled thanks to Joines and Houck's and PAMUC methods. For each of them, 10 runs were performed. First, Joines and Houck's dynamic penalty-based method was applied, but none of the 10 runs furnished a feasible solution. After about 140 generations, the algorithm converges to a solution violating the 2nd and 3rd constraints (see the illustration on a single run in Fig. 4.12), and cannot escape from it. This is due to the way dynamic penalty coefficients evolve : as their value is always increasing during the generations, when the population has not yet reached the feasible domain, the individuals with a higher level of violation of constraints are progressively discarded from the population, though they can bring useful information (by matching other individuals) to create feasible individuals. Therefore, in this case, the rigid variation of the penalties prevents the algorithm from getting away from a local (unfeasible) minimum. Only an accurate tuning of the penalties could have enabled to find an admissible solution.



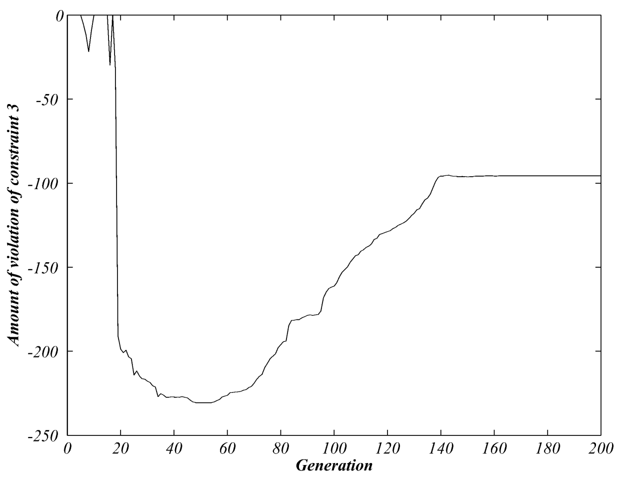


Fig. 4.12 : 2nd and 3rd constraints (sum of all individuals of the population) w.r.t. the generation for 1 run of the EA for problem S-HED (with Joines and Houck's penalty-based method).

On the contrary, all runs performed with PAMUC gave feasible solutions. The rate of feasible individuals at each generation is illustrated in Fig. 4.13, and shows that after about 50 generations, approximately 95% of the population satisfies all the constraints.

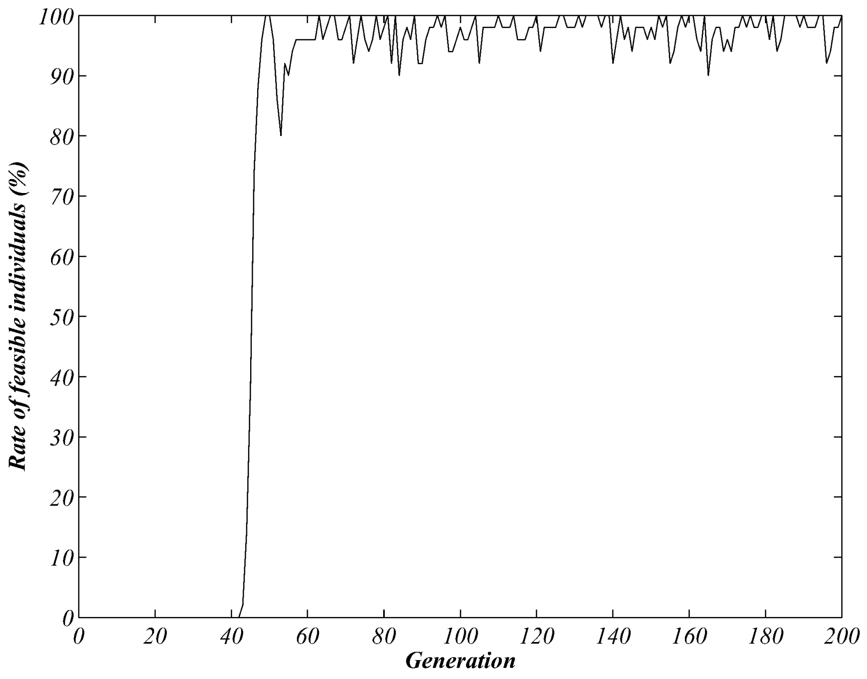


Fig. 4.13 : Rate of feasible individuals at each generation for 1 run of the EA for S-HED (with PAMUC method).

For the 10 runs of the EA, the mean of the objective function value of the best (feasible) individual (at each run) is equal to 9612.9, while the best value is 8509.8 (standard deviation = 979.4). Figure 4.14 depicts a sharp decrease of the objective function until the 80th generation, where the algorithm seems to have converged.

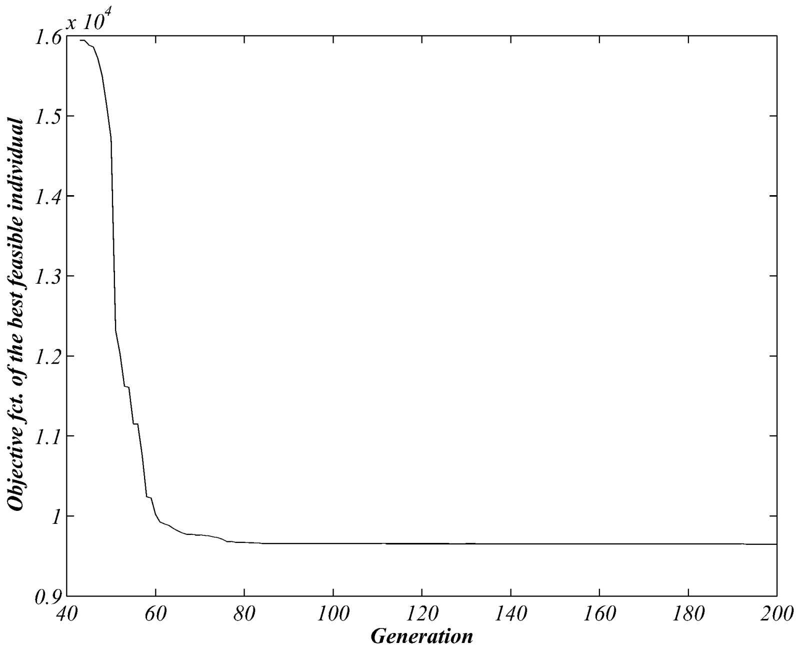


Fig. 4.14 : Objective function of the best feasible individual at each generation for 1 run of the EA (with PAMUC method) from the 43th generation for problem S-HED (the populations of the first 42 generations did not contain any feasible solution).

The results can be seriously improved by increasing the size of the population to 80 and the number of generations to 1000. Deb studied the same problem with the tournament selection method (cf. § 3.5.1.6), and the corresponding results [DEB00] are mentioned below and compared to PAMUC results (cf. Table 4.4).

Method	*Number of generations*	*Mutation*	*Niching*	*Feasible runs*	*Value of the objective function*		
					Best	*Median*	*Worst*
TS [DEB00]	*1000*	*No*	*No*	*50*	*7063.377*	*8319.211*	*13738.276*
TS [DEB00]	*1000*	*No*	*Yes*	*49*	*7065.742*	*8247.830*	*10925.165*
TS [DEB00]	*4000*	*Yes*	*Yes*	*50*	*7060.221*	*7220.026*	*10230.834*
PAMUC	*1000*	*Yes*	*No*	*50*	*7061.231*	*8193.820*	*10280.148*

Table 4.4 : Comparison of PAMUC and TS methods for problem S-HED (50 runs).

The niching used in TS method is a genetic operator intended to increase the diversity of the population by favouring the individual whose Euclidean distance with the rest of the population is greater, in order to prevent the algorithm from converging too quickly to a narrow part of the domain. It works as follows : when comparing two feasible solutions $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$, if the Euclidean distance between them is smaller than a critical distance $\bar{d}$, the solutions are compared with respect to their objective function values ; otherwise, they are not compared and another $\boldsymbol{x}^{(j)}$ is chosen. If n_f feasible individuals are checked and none is such that the corresponding Euclidean distance to $\boldsymbol{x}^{(j)}$ is greater than $\bar{d}$, $\boldsymbol{x}^{(i)}$ wins the tournament. The parameters n_f and $\bar{d}$ have to be tuned by the user.

Numerical results show that the best solution found by PAMUC is very close to the tournament selection's optimal values.

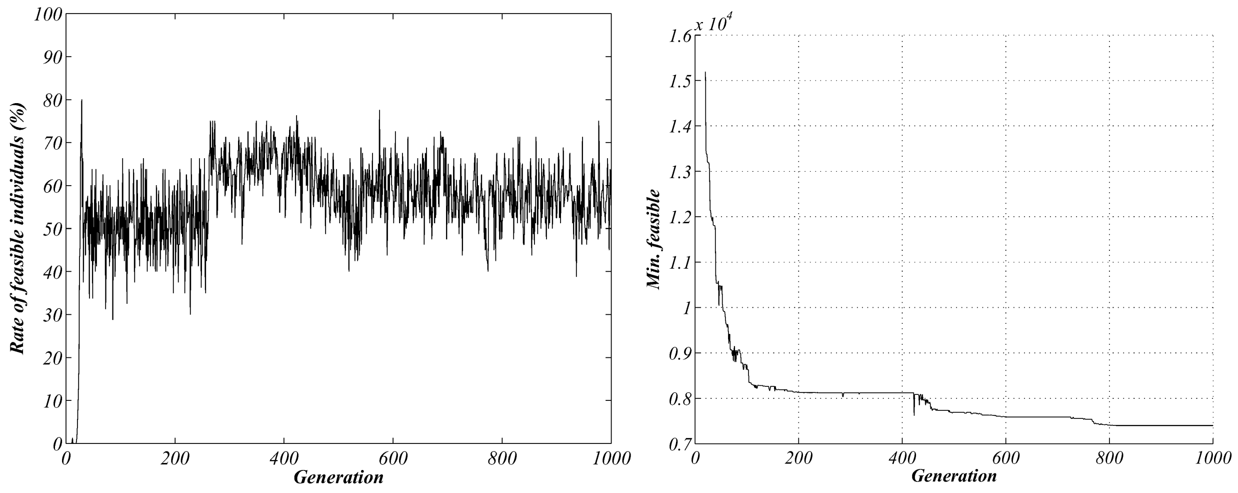


Fig. 4.15 : Rate of feasible individuals and objective function of the best feasible individual at each generation for 1 run of the EA applied to test case S-HED (with PAMUC method, $N_{gen} = 1000$).

Figure 4.15 illustrates the rate of feasible individuals and the evolution of the objective function value with respect to the generation. After the 20th generation, the rate of feasible individuals remains close to 60%, which means that a part of the population, albeit unfeasible, is very important, for it brings crucial genetic information to admissible individuals to progress towards the optimum. Adaptivity of the weights enables a continual equilibrium between the satisfaction of the constraints and the improvement of the objective function.

4.4.1.2 Test case S-3EQ

The second example is test case 80 in [HOC01] :

$$\min f(\mathbf{x}) = \exp(x_1 x_2 x_3 x_4 x_5) \tag{4.39}$$

$$\text{subject to : } h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 10, \tag{4.40}$$
$$h_2(\mathbf{x}) = x_2 x_3 - 5 x_4 x_5 = 0, \tag{4.41}$$
$$h_3(\mathbf{x}) = x_1^3 + x_2^3 = -1, \tag{4.42}$$
$$-2.3 \le x_i \le 2.3,\ i = 1, 2, \tag{4.43}$$
$$-3.2 \le x_i \le 3.2,\ i = 3, 4, 5. \tag{4.44}$$

In this example, equality constraints are transformed into inequalities as indicated in Eq. (4.45) :

$$h_j(\mathbf{x}) = 0 \ \rightarrow\ \delta - |h_j(\mathbf{x})| \ge 0 \ \text{ for } j = 1,\ldots, 3, \tag{4.45}$$

with δ set to 10^{-3} to allow some room to the EA to work. The optimal solution is [DEB00] :

$$\mathbf{x}^* = (-1.717143\ ;\ 1.595709\ ;\ 1.827247\ ;\ -0.7636413\ ;\ -0.7636450), \tag{4.46}$$

with an objective function value $f(\mathbf{x}^*) = 0.053950$.

Joines and Houck's method (implemented in the Std-EA) and PAMUC have been used ; the EA parameters of the study are collected in Table 4.5.

Symbol	*Parameter*	*Value*
Env	*Environment*	*Std-EA Matlab*
Cod	*Coding of the variables*	*Real coding*
N	*Size of the population*	*50*
N_{gen}	*Number of generations*	*1000*
T_s	*Type of selection*	*Tournament*
n_t	*Number of individuals participating to a tournament*	*2*
p_c	*Probability of crossover*	*1*
T_c	*Type of crossover*	*SBX*
p_m	*Probability of mutation*	*Variable**
T_m	*Type of mutation*	*PBM*
η_c	*Distribution index for crossover*	*1*
η_m	*Distribution index for mutation*	*Variable**

Table 4.5 : EA parameters for test case S-3EQ (cf. § 2.3.2.6).*

While 13 runs (of 20) of the EA combined with Joines and Houck's method furnished a feasible point, each run performed with PAMUC found a solution satisfying the whole set of constraints. Furthermore, 7 runs found the global optimum $\boldsymbol{x}^*$.

The corresponding results are mentioned in Table 4.6. Figures 4.16 and 4.17 depict the rate of constraint satisfaction of the population, as well as the evolution of the objective function value of the best feasible individual, at each generation (for one run of the EA combined respectively with Joines and Houck's and PAMUC methods). Results obtained by Joines and Houck's technique show that the rate of feasible members in the population oscillates from larger periods where it is very low to peaks of high level of admissibility ; this explains the discontinuous shape of the best feasible objective function (indeed, periods where no admissible solution is generated are of course not represented). On the contrary, with PAMUC, the number of feasible individuals remains globally beyond a level of 50%. For both methods, it appears that using a large number of generations leads to an improvement of the solution. One can suppose by considering Fig. 4.17 [right] that using N_{gen} larger than 1000 would have certainly improved the solution. Nevertheless, a fixed value of N_{gen} (= 1000) was used to compare PAMUC and Joines and Houck's techniques with the same number of function evaluations.

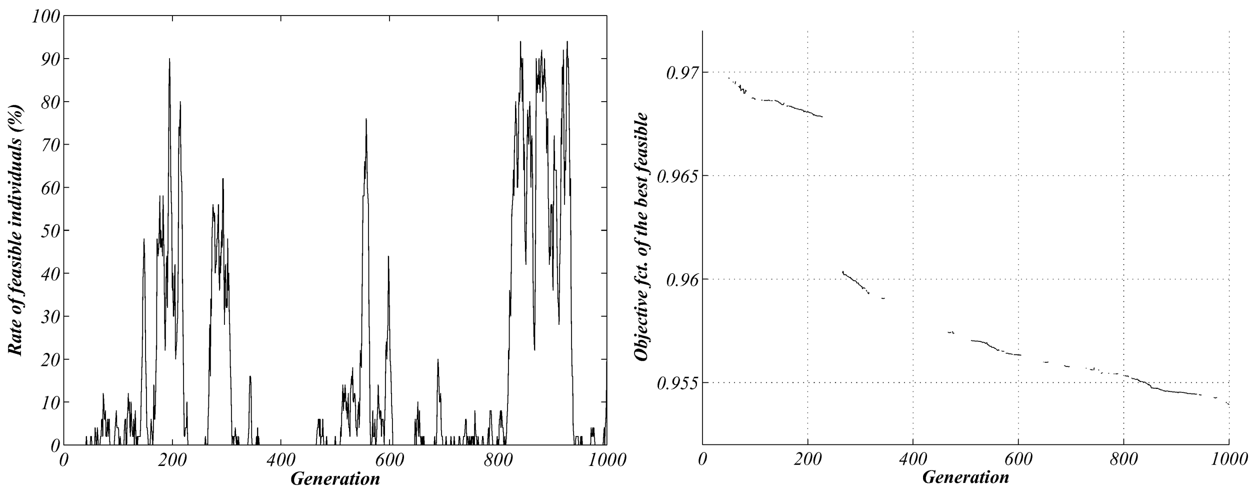
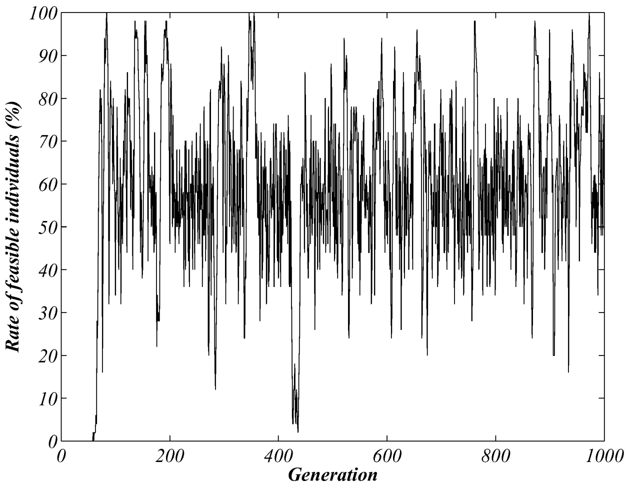


Fig. 4.16 : Rate of feasible individuals and objective function of the best feasible individual at each generation for 1 run of the EA applied to test case S-3EQ (with Joines and Houck's method implemented in the Std-EA).

Method	*Feasible runs (over 20)*	*Value of the objective function*	
		Best	*Mean*
Joines and Houck	*13*	*0.35432*	*0.87655*
PAMUC	*20*	*0.05395*	*0.35433*

Table 4.6 : Results of Joines and Houck's method (implemented in the Std-EA) and PAMUC for S-3EQ.



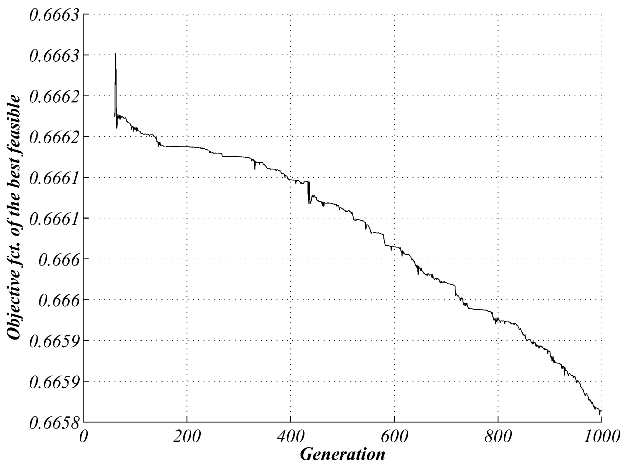


Fig. 4.17 : Rate of feasible individuals and objective function of the best feasible individual at each generation for 1 run of the EA applied to test case S-3EQ (with PAMUC method).

4.4.1.3 Test case S-6ACT

The third example is test case 113 in [HOC01], and is defined by Eqs. (4.47) to (4.56) :

$$\min f(\mathbf{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14 x_1 - 16 x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5 x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \quad (4.47)$$

$$\text{subject to : } g_1(\mathbf{x}) = 105 - 4 x_1 - 5 x_2 + 3 x_7 - 9 x_8 \geq 0, \quad (4.48)$$

$$g_2(\mathbf{x}) = -10 x_1 + 8 x_2 + 17 x_7 - 2 x_8 \geq 0, \quad (4.49)$$

$$g_3(\mathbf{x}) = 8 x_1 - 2 x_2 + 17 x_7 - 2 x_8 \geq 0, \quad (4.50)$$

$$g_4(\mathbf{x}) = -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2 x_3^2 + 7 x_4 + 120 \geq 0, \quad (4.51)$$

$$g_5(\mathbf{x}) = -5 x_1^2 - 8 x_2 - (x_3 - 6)^2 + 2 x_4 + 40 \geq 0, \quad (4.52)$$

$$g_6(\mathbf{x}) = -x_1^2 - 2(x_2 - 2)^2 + 2 x_1 x_2 - 14 x_5 + 6 x_6 \geq 0, \quad (4.53)$$

$$g_7(\mathbf{x}) = -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3 x_5^2 + x_6 + 30 \geq 0, \quad (4.54)$$

$$g_8(\mathbf{x}) = 3 x_1 - 6 x_2 - 12(x_9 - 8)^2 + 7 x_{10} \geq 0, \quad (4.55)$$

$$-10 \leq x_i \leq 10, \; i = 1, \ldots, 10. \quad (4.56)$$

The optimal solution is [DEB00] :

$$\mathbf{x}^* = (2.171996 \; ; 2.363683 \; ; 8.773926 \; ; 5.095984 \; ; 0.9906548 \; ; 1.430574 \; ; 1.321644 \; ; 9.828726 \; ; 8.280092 \; ; 8.375927), \quad (4.57)$$

with an objective function value $f(\mathbf{x}^*) = 24.3062091$.

Joines and Houck’s and PAMUC methods have been used ; the EA parameters of the study are collected in Table 4.7.

Symbol	***Parameter***	***Value***
Env	*Environment*	*Std-EA Matlab*
Cod	*Coding of the variables*	*Real coding*
N	*Size of the population*	*100*
N_{gen}	*Number of generations*	*100*
T_s	*Type of selection*	*Tournament*
n_t	*Number of individuals participating to a tournament*	*2*
p_c	*Probability of crossover*	*1*
T_c	*Type of crossover*	*SBX*
p_m	*Probability of mutation*	*Variable**
T_m	*Type of mutation*	*PBM*
η_c	*Distribution index for crossover*	*1*
η_m	*Distribution index for mutation*	*Variable**

Table 4.7 : EA parameters for test case S-6ACT (cf. § 2.3.2.6).*

Table 4.8 presents the performance of the EA with Joines and Houck’s and PAMUC methods, showing better results for PAMUC.

Method	***Feasible runs***	***Value of the objective function***		
		Best	***Mean***	***Std deviation***
Joines and Houck	*20*	*35.9251*	*86.6529*	*39.9731*
PAMUC	*20*	*33.7940*	*74.2710*	*33.9380*

Table 4.8 : Comparison of Joines and Houck’s method (implemented in the Std-EA) and PAMUC for problem S-6ACT (20 runs).

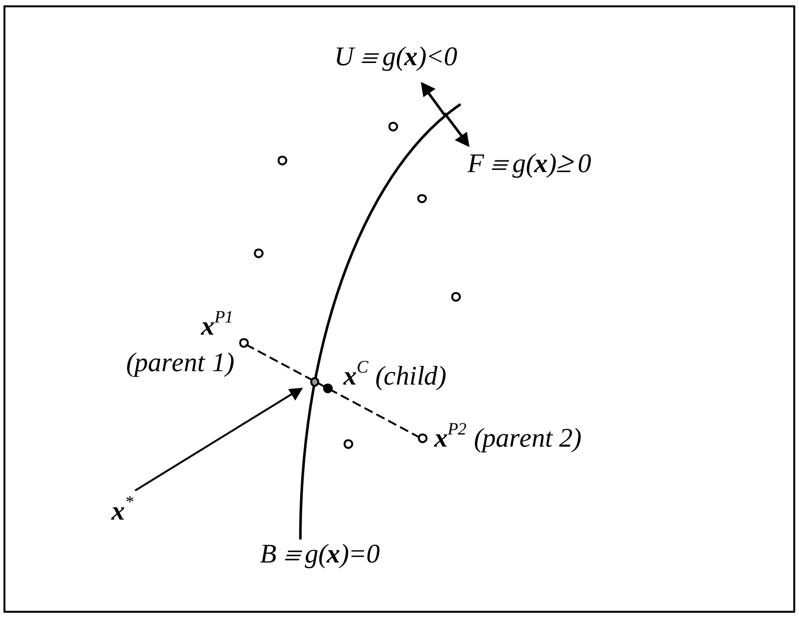

$U \equiv g(\mathbf{x})<0$
$F \equiv g(\mathbf{x})\geq 0$
$\mathbf{x}^{P1}$
(parent 1)
$\mathbf{x}^{C}$ (child)
$\mathbf{x}^{P2}$ (parent 2)
$\mathbf{x}^{*}$
$B \equiv g(\mathbf{x})=0$

Fig. 4.18 : Boundary (B) between feasible (F) and unfeasible (U) points in the variable space for a 1-constraint problem, and illustration of crossover between an unfeasible parent ($\mathbf{x}^{P1}$) and a feasible parent ($\mathbf{x}^{P2}$), creating a child closer to the boundary B (with $\mathbf{x}^{*}$ being the global optimum).

It should be noticed that in this example, the first 6 constraints are active at the optimum, i.e. $g_i(\mathbf{x}^*) = 0$ for $i = 1, \dots, 6$. In this case, using adaptive weights is a fruitful option, as depicted in Fig. 4.18. Indeed, as unfeasible individuals are accepted in the population with PAMUC (unlike some other constraint-handling techniques, as the death penalty [cf. § 3.5.1.1], which discard unfeasible points), when one (or several) constraint(s) is (are) active at the optimum (i.e. $g_i(\mathbf{x}^*) = 0$ for at least one i), they can generate – by matching feasible points – new individuals closer to the boundary separating admissible and inadmissible domains, on which lies the optimal solution $\mathbf{x}^*$.

4.4.1.4 Test case S-CRES

This is the first test case proposed by Deb in [DEB00]. It is characterized by 2 constraints which reduce the feasible domain F to approximatively 0.7% of the total search space (cf. the crescent shape of F in Fig. 4.19 zoomed in Fig. 4.20). The problem is defined as follows :

$$\min f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{4.58}$$

$$\text{subject to : } g_1(\mathbf{x}) = 4.84 - (x_1 - 0.05)^2 - (x_2 - 2.5)^2 \geq 0, \tag{4.59}$$

$$g_2(\mathbf{x}) = x_1^2 + (x_2 - 2.5)^2 - 4.84 \geq 0, \tag{4.60}$$

$$0 \leq x_1 \leq 6 \text{ ; } 0 \leq x_2 \leq 6. \tag{4.61}$$

The constrained optimum solution is [DEB00] :

$$\mathbf{x}^* = (2.246826 \text{ ; } 2.381865) \text{ with } f(\mathbf{x}^*) = 13.59085. \tag{4.62}$$

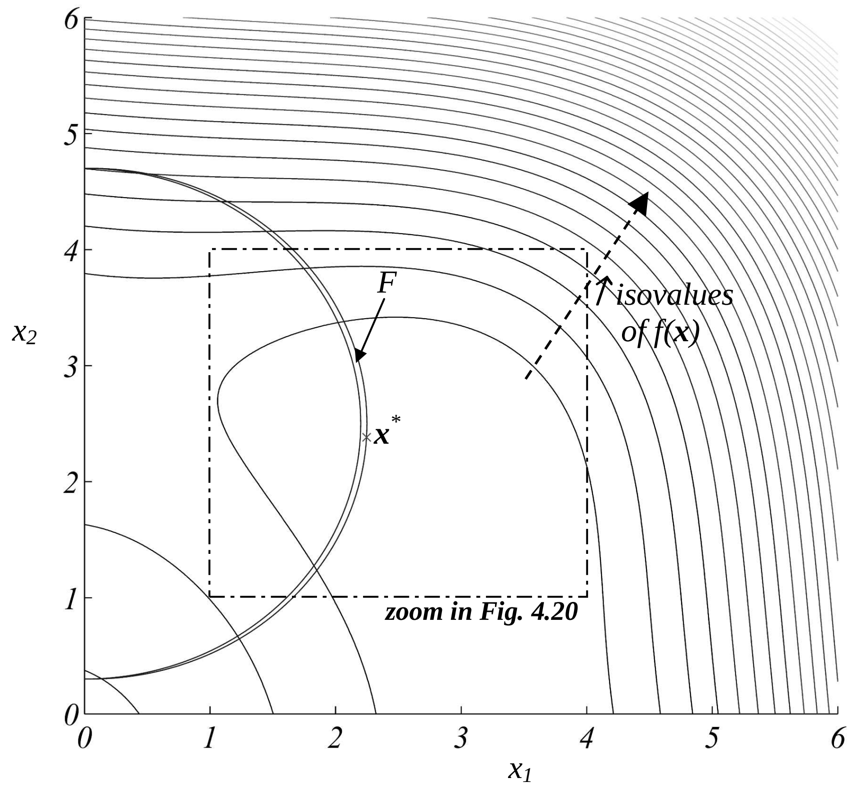


Fig. 4.19 : Crescent shape of the feasible domain F in test case S-CRES in the variable space (x_1 , x_2) (contours are isovalues of the objective function f) (figure adapted from [DEB00]).

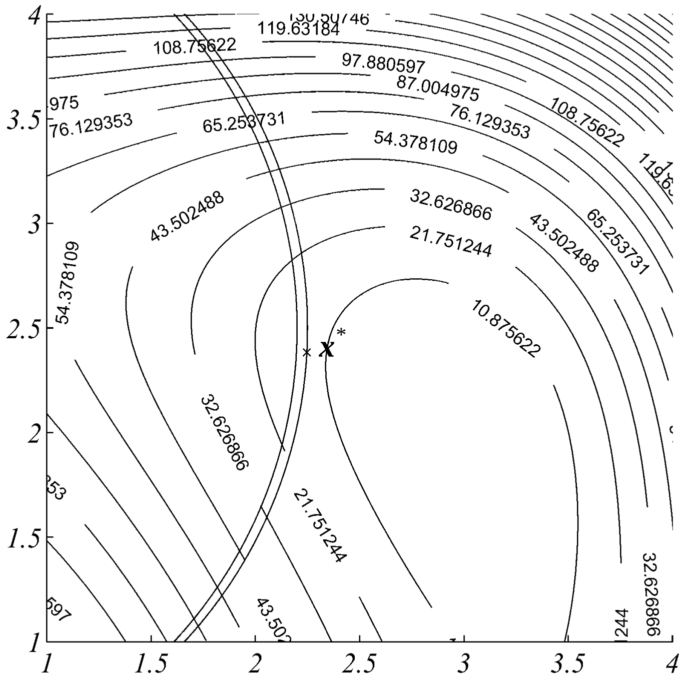


Fig. 4.20 : Zoom of the feasible domain F in test case S-CRES in the variable space (x_1, x_2) in the vicinity of the global feasible optimum $\mathbf{x}^*$ (figure adapted from [DEB00]).

Parameters of the study	***PS (R=0.01) [DEB00]***	***PS (R=1) [DEB00]***	***TS [DEB00]***	***PAMUC***
Number of feasible runs (of 50)	31	39	50	50
Best feasible solution	13.58958	13.59108	13.59085	13.59104
Median feasible solution	24.07437	16.35284	13.61673	13.65329
Worst feasible solution	114.69033	172.81369	117.02971	118.45128

Table 4.9 : Results for test case S-CRES for 50 runs.

In Table 4.9, solutions obtained with PAMUC are compared to results given in [DEB00] for PS and TS methods discussed above (cf. § 4.2.2). The following parameters are used : real coding of the variables, binary tournament and simulated binary crossover.

It should be noted now that the distribution index η_c, which controls the SBX operator (see § 2.3.2.5) with real coding, is equal to 1 in all cases treated in this work. As described in [DEB95], a small value of η_c ($\eta_c \ll 1$) allows solutions far away from parents to be created, whereas a large value of η_c ($\eta_c \gg 1$) restricts only near-parent solutions to be created as children. The crossover probability p_c is equal to 0.9, and no mutation is performed ; the number of generations $N_{gen} = 50$ and the size of the population $N = 20$.

In Fig. 4.21, the average Euclidean norm of the variables (compared to the theoretical solution $\mathbf{x}^*$) illustrates the convergence of PAMUC towards the global optimum. Table 4.9 shows that PAMUC and TS methods both provide very close results.

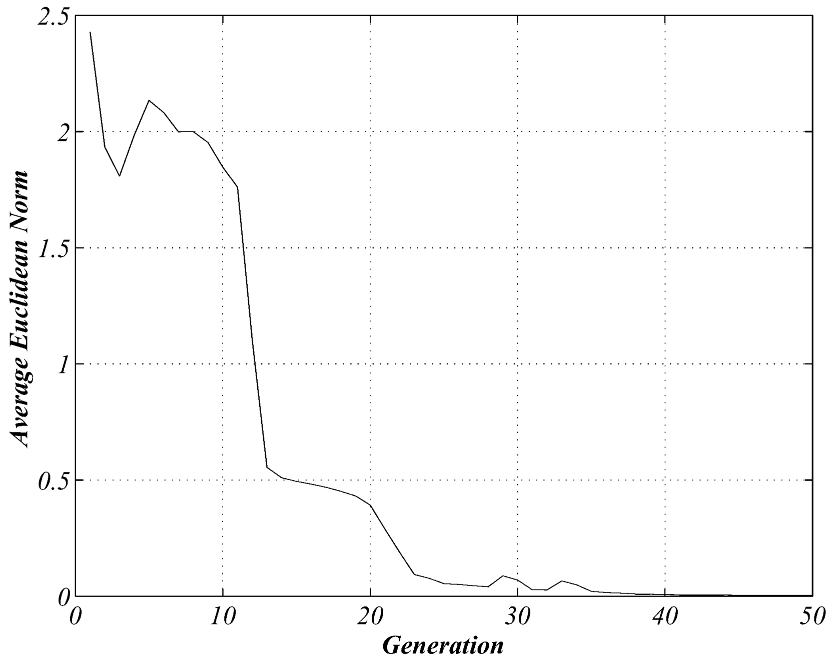


Fig. 4.21 : Average Euclidean norm of the feasible individuals of the population at each generation for test case S-CRES (with PAMUC).

4.4.1.5 Test case S-38IC

This problem has 5 variables and a high number of constraints (38 inequality constraints ; cf. test case 2 in [DEB00]).

$$\min f(\mathbf{x}) = 0.1365 - 5.843.10^{-7} y_{17} + 1.17.10^{-4} y_{14} + 2.358.10^{-5} y_{13} + 1.502.10^{-6} y_{16} + 0.0321\, y_{12} + 0.004324\, y_5 + 10^{-4}\, c_{15}/c_{16} + 37.48\, y_2/c_{12} \tag{4.63}$$

$$\text{subject to : } g_1(\mathbf{x}) = 1.5\, x_2 - x_3 \geq 0, \tag{4.64}$$

$$g_2(\mathbf{x}) = y_1(\mathbf{x}) - 213.1 \geq 0, \tag{4.65}$$

$$g_3(\mathbf{x}) = 405.23 - y_1(\mathbf{x}) \geq 0, \tag{4.66}$$

$$g_{j+2}(\mathbf{x}) = y_j(\mathbf{x}) - a_j \geq 0 \quad \text{for } j = 2,\ldots, 17, \tag{4.67}$$

$$g_{j+18}(\mathbf{x}) = b_j - y_j(\mathbf{x}) \geq 0 \quad \text{for } j = 2,\ldots, 17, \tag{4.68}$$

$$g_{36}(\mathbf{x}) = y_4(\mathbf{x}) - 0.28/0.72\, y_5(\mathbf{x}) \geq 0, \tag{4.69}$$

$$g_{37}(\mathbf{x}) = 21 - 3496.0\, y_2(\mathbf{x})/c_{12}(\mathbf{x}) \geq 0, \tag{4.70}$$

$$g_{38}(\mathbf{x}) = 62212.0/c_{17}(\mathbf{x}) - 110.6 - y_1(\mathbf{x}) \geq 0, \tag{4.71}$$

$$704.4148 \leq x_1 \leq 906.3855, \tag{4.72}$$

$$68.6 \leq x_2 \leq 288.88, \tag{4.73}$$

$$0 \leq x_3 \leq 134.75, \tag{4.74}$$

$$193 \leq x_4 \leq 287.0966, \tag{4.75}$$

$$25 \leq x_5 \leq 84.1988. \tag{4.76}$$

The terms $y_j(\mathbf{x})$ and the parameters a_j and b_j are defined in Appendix A. The optimal solution found in [DEB00] is :

$$x^* = (707.337769 ; 68.600273 ; 102.900146 ; 282.024841 ; 84.198792), \quad (4.77)$$

with the corresponding value of the objective function $f(x^*)$ equal to – 1.91460.

At this solution, none of the constraints is active (i.e. the solution lies inside the feasible domain). In Table 4.10, solutions obtained with PAMUC are compared to results given in [DEB00] for PS-a (with parameter a such that $R = 10^a$: cf. § 3.5.1.6) and TS methods. The parameters used in this example are the same as in the previous case, except that the number of generations $N_{gen} = 1000$ and the size of the population $N = 50$.

Parameters of the study	***PS-0 [DEB00]***	***PS-2 [DEB00]***	***PS-6 [DEB00]***	***TS [DEB00]***	***TS [DEB00]***	***PAMUC***
Mutation	No	No	No	No	Yes	Yes
Niching	No	No	No	Yes	Yes	No
Number of runs (out of 50) which found a feasible solution	12	50	50	50	50	50
Best feasible solution	– 1.86365	– 1.89845	– 1.91319	– 1.91410	– 1.91460	– 1.90563
Median feasible solution	– 1.69507	– 1.65156	– 1.65763	– 1.85504	– 1.91457	– 1.88615
Worst feasible solution	– 1.35910	– 1.00969	– 1.11550	– 1.30643	– 1.91454	– 1.42366

Table 4.10 : Results for problem S-38IC for 50 runs.

Results of PAMUC are illustrated on Fig. 4.22 and 4.23. Fig. 4.22 [left] shows the fast convergence of the population towards the feasible domain, and Fig. 4.22 [right] depicts the convergence to the feasible optimum x^*. The average Euclidean norm on the variables (compared to the theoretical solution) also illustrates the convergence towards x^* (see Fig. 4.23). Table 4.10 shows that results obtained by PAMUC are better than those obtained by PS-a, except for $a = 6$ ($R = 10^6$), but PAMUC has provided good result without requiring any tuning of parameters. TS method outperforms both methods (PS and PAMUC).

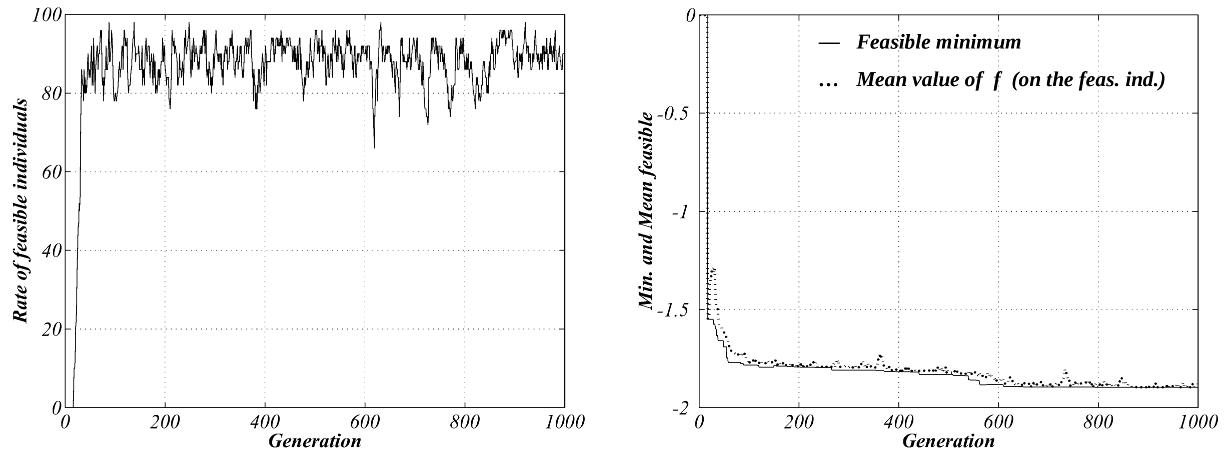


Fig. 4.22 : Rate [left] and minimum and mean value of the objective function [right] of feasible individuals of the population at each generation for one run for test case S-38IC (with PAMUC).

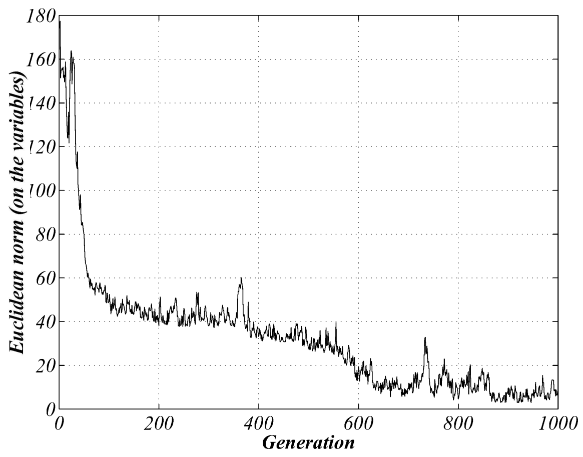


Fig. 4.23 : Average Euclidean norm (on the variables) of the feasible individuals of the population at each generation for one run for test case P-IC38 (with PAMUC).

4.4.1.6 Test case S-0.5F

This problem has 7 variables and 4 nonlinear constraints (test case 5 in [DEB00]).

$$\min f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10\, x_5^6 + 7\, x_6^2 + x_7^4 - 4\, x_6 x_7 \tag{4.78}$$

$$\text{subject to : } g_1(\mathbf{x}) = 127 - 2\, x_1^2 - 3\, x_2^4 - x_3 - 4\, x_4^2 - 5\, x_5 \geq 0, \tag{4.79}$$

$$g_2(\mathbf{x}) = 282 - 7\, x_1 - 3\, x_2 - 10\, x_3^2 - x_4 + x_5 \geq 0, \tag{4.80}$$

$$g_3(\mathbf{x}) = 196 - 23\, x_1 - x_2^2 - 6\, x_6^2 + 8\, x_7 \geq 0, \tag{4.81}$$

$$g_4(\mathbf{x}) = -4\, x_1^2 - x_2^2 + 3\, x_1 x_2 - 2\, x_3^2 - 5\, x_6 + 11\, x_7 \geq 0, \tag{4.82}$$

$$-10 \leq x_i \leq 10 \text{ for } i = 1,\ldots, 7. \tag{4.83}$$

The optimal solution is :

$$\mathbf{x}^* = (2.330499 ; 1.951372 ; -0.4775414 ; -0.6244870 ; 1.038131 ; 1.594227), \tag{4.84}$$

with the corresponding value of the objective function f^* equal to 680.63. Only about 0.5% of the search space is feasible [DEB00]. The parameters used in the study are the same as in the previous example, save that the size of the population N = 70. In Table 4.11, solutions obtained with PAMUC are compared to results given in [DEB00]. It is interesting to note that the presence of niching (in TS method with the critical distance $\bar{d}$ = 0.1 and n_f = 0.25 N : cf. § 4.4.1.1) or mutation (in PAMUC) enables to find better solutions by creating diversity among the population.

Parameters of the study	***TS [DEB00]***	***TS [DEB00]***	***PAMUC***	***PAMUC***
Mutation	No	No	No	Yes
Niching	No	Yes	No	No
Number of runs (out of 50) which found a feasible solution	50	50	50	50
Best feasible solution	680.800720	680.659424	680.810394	680.729460
Median feasible solution	683.076843	681.525635	684.139966	682.344782
Worst feasible solution	705.861145	687.188599	706.238773	689.122543

Table 4.11 : Results for problem S-0.5F for 50 runs.

4.4.1.7 Test case S-HIM

The last single-objective problem taken from [DEB00] is Himmelblau's nonlinear optimization problem. This problem has 5 variables and 6 inequality constraints (test case 6 in [DEB00]) :

$$\min f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1 x_5 + 37.293239\, x_1 - 40792.141 \tag{4.85}$$

$$\text{subject to : } g_1(\mathbf{x}) = 85.334407 + 0.0056858\, x_2 x_5 + 0.0006262\, x_1 x_4 - 0.0022053\, x_3 x_5 \geq 0, \tag{4.86}$$

$$g_2(\mathbf{x}) = 92 - 85.334407 - 0.0056858\, x_2 x_5 - 0.0006262\, x_1 x_4 + 0.0022053\, x_3 x_5 \geq 0, \tag{4.87}$$

$$g_3(\mathbf{x}) = 80.51249 + 0.0071317x_2 x_5 + 0.0029955x_1 x_2 + 0.0021813x_3^2 - 90 \geq 0, \tag{4.88}$$

$$g_4(\mathbf{x}) = 110 - 80.51249 - 0.0071317x_2 x_5 - 0.0029955x_1 x_2 - 0.0021813x_3^2 \geq 0, \tag{4.89}$$

$$g_5(\mathbf{x}) = 9.300961 + 0.0047026\, x_3 x_5 + 0.0012547\, x_1 x_3 + 0.0019085\, x_3 x_4 - 20 \geq 0, \tag{4.90}$$

$$g_6(\mathbf{x}) = 25 - 9.300961 - 0.0047026\, x_3 x_5 - 0.0012547\, x_1 x_3 - 0.0019085\, x_3 x_4 \geq 0, \tag{4.91}$$

$$78 \leq x_1 \leq 102, \tag{4.92}$$

$$33 \leq x_2 \leq 45, \tag{4.93}$$

$$27 \leq x_i \leq 45 \text{ for } i = 3,\ldots, 5. \tag{4.94}$$

The parameters used with PAMUC are the same as in the previous example, except that the number of generations N_{gen} = 100 and the size of the population N = 50. Both penalty-based methods used a GA with a binary representation, two-point crossover, tournament selection and uniform mutation.

Parameters of the study	***Static penalty (Homaifar et al.)***	***Dynamic penalty (Joines & Houck with C = 0.5 and $\alpha = \beta = 2$)***	***MGA (Coello)***	***PAMUC***
Crossover probability p_c	0.8	0.8	self-adapted	0.8
Mutation probability p_m	0.005	0.005	self-adapted	dynamic
Best feasible solution	– 30790.27159	– 30903.877	– 31005.7966	– 30946.2155
Mean feasible solution	– 30446.4618	– 30539.9156	– 30862.8735	– 30598.6437
Worst feasible solution	– 29834.3847	– 30106.2498	– 30721.0418	– 30310.6035
Standard deviation	226.3428	200.035	73.240	163.718

Table 4.12 : Results for problem S-HIM for 50 runs.

The PAMUC method provided very good results with the same number of fitness function evaluations, compared to penalty-based methods, while the MGA (multi-objective genetic algorithm), based on a nondominance approach [COE00a], gives better results (see Table 4.12). It can also be mentioned that the coevolutionary penalty method proposed by Coello [COE02], which has furnished the best solution so far for this problem, needs a considerably higher number of function evaluations (900 000 instead of 5000 in results cited in Table 4.12).

4.4.1.8 Test case P-WBD

The last single-objective test case is the welded beam design problem, formulated as follows [COE00a] (see Fig. 4.24) :

$$\min f(\mathbf{x}) = 1.10471\, x_1^2\, x_2 + 0.04811\, x_3\, x_4\, (14 + x_2) \tag{4.95}$$

$$\text{subject to : } g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{max} \le 0, \tag{4.96}$$

$$g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{max} \le 0, \tag{4.97}$$

$$g_3(\mathbf{x}) = x_1 - x_4 \le 0, \tag{4.98}$$

$$g_4(\mathbf{x}) = 0.10471\, x_1^2 + 0.04811\, x_3\, x_4\, (14 + x_2) - 5 \le 0, \tag{4.99}$$

$$g_5(\mathbf{x}) = 0.125 - x_1 \le 0, \tag{4.100}$$

$$g_6(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{max} \le 0, \tag{4.101}$$

$$g_7(\mathbf{x}) = P - P_c(\mathbf{x}) \le 0, \tag{4.102}$$

$$\text{where : } \tau(\mathbf{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \tag{4.103}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tag{4.104}$$

$$\tau'' = \frac{MR}{J}, \tag{4.105}$$

$$M = P\left(L + \frac{x_2}{2}\right), \tag{4.106}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \tag{4.107}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \tag{4.108}$$

$$\sigma(\mathbf{x}) = \frac{6PL}{x_4x_3^2}, \tag{4.109}$$

$$\delta(\mathbf{x}) = \frac{4PL^3}{Ex_3^3x_4}, \tag{4.110}$$

$$P_c(\mathbf{x}) = \frac{4.013\, E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \tag{4.111}$$

expressions in which P = 6000 lb, L = 14 in, E = 30.10^6 psi, G = 12.10^6 psi, τ_{max} = 13600 psi, σ_{max} = 30000 psi and δ_{max} = 0.25 in.

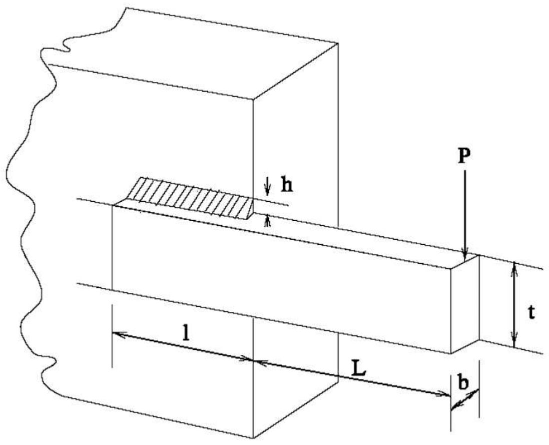


Fig. 4.24 : Welded beam design problem [COE00a].

Joines and Houck's method (implemented in the Std-EA with $C = 1$, $\alpha = 1$ and $\beta = 2$) and PAMUC were used ; the EA parameters of the study are collected in Table 4.13. Table 4.14 presents the performance of the EA with both methods, showing better results for PAMUC. Once again PAMUC outperforms Joines and Houck's method.

Symbol	*Parameter*	*Value*
Env	*Environment*	*Std-EA Matlab*
Cod	*Coding of the variables*	*Gray binary coding*
N_{bit}	*Numbe of bits per variable*	*8*
N	*Size of the population*	*80*
N_{gen}	*Number of generations*	*150*
T_s	*Type of selection*	*Tournament*
n_t	*Number of individuals participating to a tournament*	*2*
p_c	*Probability of crossover*	*1*
T_c	*Type of crossover*	*2-point*
p_m	*Probability of mutation*	*0.01*
T_m	*Type of mutation*	*Flip*

Table 4.13 : EA parameters for test case P-WBD (cf. § 2.3.2.6).*

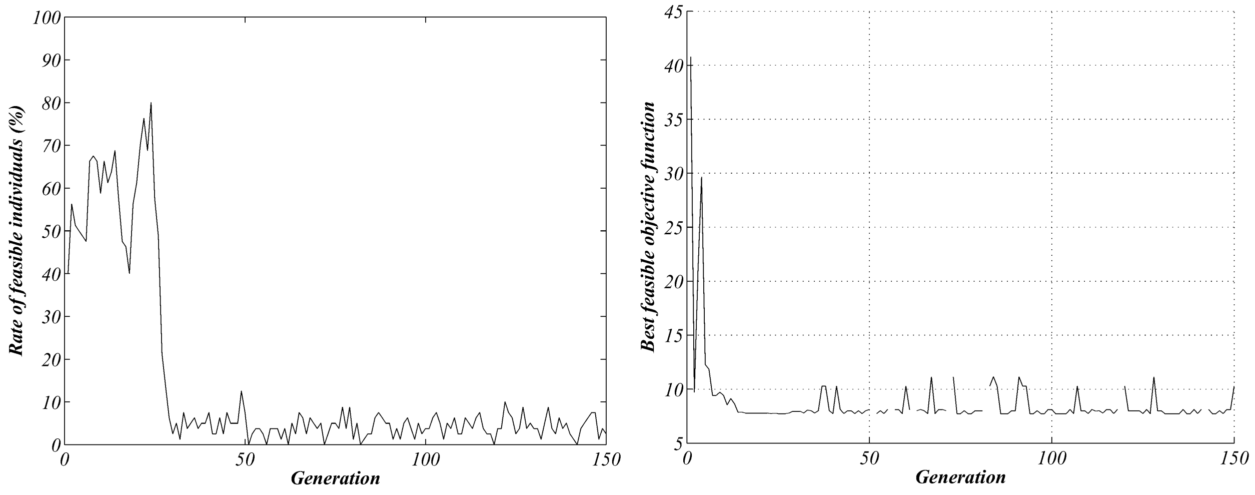


Fig. 4.25 : Rate of feasible individuals and objective function of the best feasible individual at each generation for 1 run of the EA applied to test case P-WBD (with Joines and Houck's method).

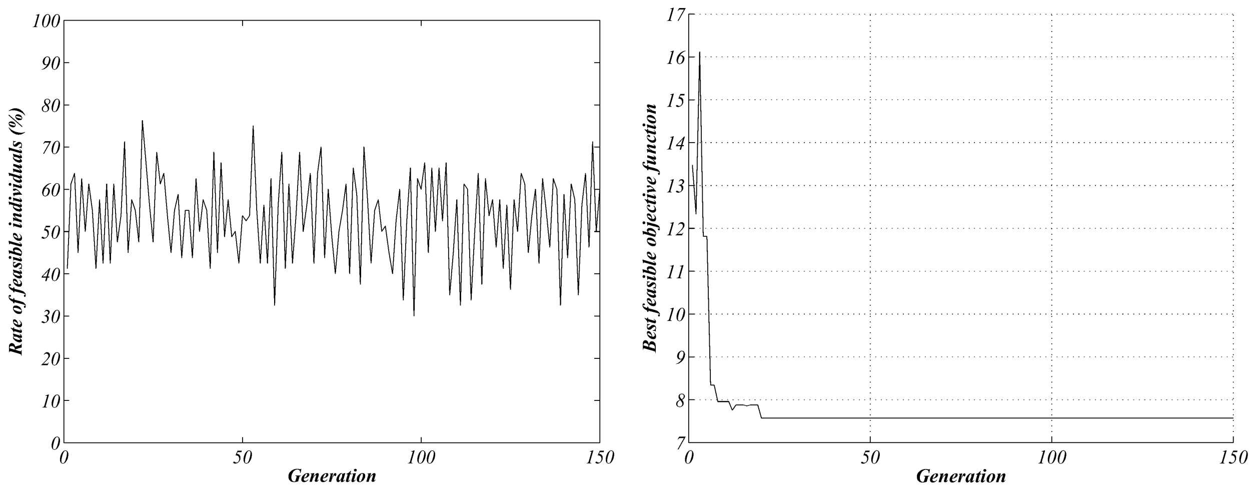


Fig. 4.26 : Rate of feasible individuals and objective function of the best feasible individual at each generation for 1 run of the EA applied to test case P-WBD (with PAMUC).

Method	Feasible runs	Value of the objective function		
		Best	Mean	Std deviation
Joines and Houck	50	7.439418	8.313480	0.904600
PAMUC	50	8.785876	9.270780	0.484904

Table 4.14 : Comparison of Joines and Houck's and PAMUC results for problem P-WBD (50 runs).

Fig. 4.25 and 4.26 show that with Joines and Houck's method, after a boom, the rate of feasible members of the population decreases drastically, whilst in PAMUC results it oscillates around 50%.

4.4.1.9 Remarks on single-objective constrained problems

A recapitulatory table gathers the results of PAMUC compared to other methods investigated in this study :

Test case (SOCO)	Name of the problem	PAMUC	Joines and Houck	PS [DEB00]	TS [DEB00]
1	S-HED	7061.231	(no feasible solution)	(results not available)	7065.742
2	S-3EQ	0.05395	0.35432	(results not available)	(results not available)
3	S-6ACT	33.7940	35.9251	(results not available)	(results not available)
4	S-CRES	13.59104	(results not available)	13.58958	13.59085
5	S-38IC	– 1.90563	(results not available)	– 1.91319	– 1.91460
6	S-0.5F	680.729460	(results not available)	(results not available)	680.659424
7	S-HIM	– 30946.2155	– 30903.877	(results not available)	– 31005.7966
8	P-WBD	7.439418	8.785876	(results not available)	(results not available)

Table 4.15 : Best feasible solutions found by PAMUC and other methods mentioned in this study (for a same number of function evaluations).

Although PAMUC was dedicated to multiobjective (constrained) problems, results mentioned in Table 4.15 show that it provides very good results compared to methods specially devoted to single-objective contrained optimization. This is mainly due to the adaptivity of the weights, which seems to be more efficient than a “blind” evolution of the penalty coefficients, which does not take into account the results of the EA during the generations (in terms of rate of feasible individuals in the population). Furthermore, PAMUC does not require any parameter tuning, unlike most penalty-based techniques.

However, it should be underlined that when one is only interested in solving single-objective problems, Deb’s tournament selection technique seems the most appropriate approach, since it generally gives better results than PAMUC for the same number of function evaluations, and for less computational overhead (in comparison with the use of PROMETHEE II in PAMUC). Moreover, for arduous problems, Coello’s coevolutionary method [COE02] is the best way to reach the feasible optimal solution, though it needs a considerably higher number of function evaluations. Finally, as already mentioned in §§ 3.5.1.3 and 3.5.1.4, for some specific applications, the use of decoders or repair strategy (related to the coding of the variables or the definition of the genetic operators) also provide very satisfactory results in tackling the constraints.

Now that the ability of PAMUC to deal with constraints in single-objective examples has been demonstrated, the next section will present multiobjective problems.

4.4.2 Multiobjective optimization (MOO)

4.4.2.1 Test case M-UC

The first multiobjective example is an unconstrained problem taken from Cvetković [CVE00] and is characterized by 2 variables and 2 objective functions :

$$max \begin{cases} f_1(\mathbf{x}) = sin(x_1^{\,2} + x_2^{\,2} - 1) & (4.112) \\ f_2(\mathbf{x}) = sin(x_1^{\,2} + x_2^{\,2} + 1) & (4.113) \end{cases}$$

$$with\ \ 0 \le x_1 \le 3\pi/4\ \ and\ \ 0 \le x_2 \le 3\pi/4. \qquad (4.114)$$

The procedure consisting in varying the weights in order to check that corresponding results correctly depict the user’s preferences is illustrated in Fig. 4.27, which shows different points (in the objective space) obtained by PAMUC for 8 values of the weights. The EA used here is the same as in test case S-CRES (cf. § 3.4.1.4) save that N_{gen} = 40 and N = 40.

To analyze PAMUC more rigorously and compare it to the weighted sum method (WS), the following procedure is applied : 20 processes were launched, with each process consisting in running the EA with both methods (PAMUC and the WS) 30 times, with a set of weights varying from $\{w_1^* = 0 ; w_2^* = 1\}$ for the first run to $\{w_1^* = 1 ; w_2^* = 0\}$ for the last run, by a constant step.

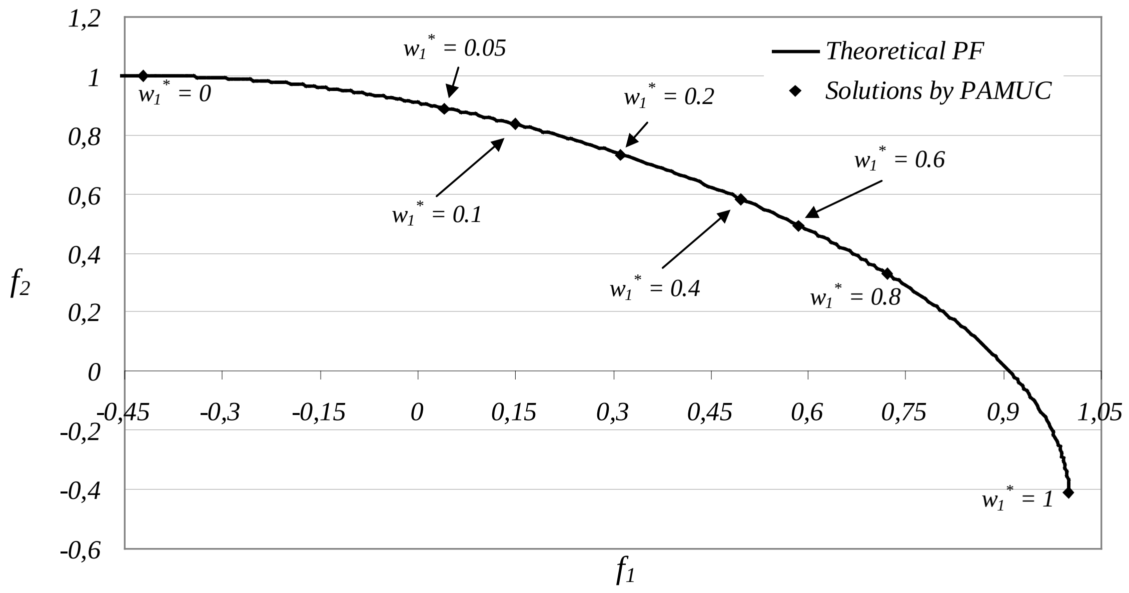


Fig. 4.27 : Nondominated solutions obtained by PAMUC method for M-UC (for 8 values of the weights, with $w_2^* = 1 - w_1^*$).

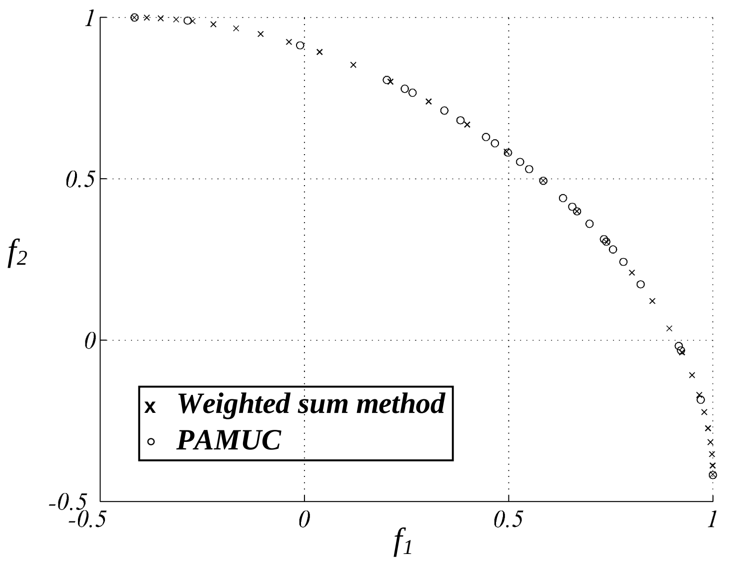


Fig. 4.28 : Nondominated solutions obtained with the weighted sum and PAMUC methods for M-UC (for one process).

Statistics on the 20 processes show that the mean value for *R1(PAMUC,WS)* is equal to 0.74 (with a standard deviation of 0.28), which should mean that PAMUC outperforms the weighted sum method (see § 4.3.5). However, Fig. 4.28 (illustrating nondominated solutions obtained by both methods for one process) does not confirm this statement : the results obtained by both methods seem quite similar. As a matter of fact, performing the computations with much higher numbers of processes (1000 instead of 20) and of runs by process (100 instead of 30) shows that PAMUC and the weighted sum method are almost equivalent, since the mean of *R1(PAMUC,WS)* is equal to 0.51 (see Table 4.16). The misleading results obtained with only 20 processes illustrate some weakness of R1-norm to provide meaningful information when the nondominated sets to be compared are very close one to another. Mathematically speaking, this can be easily explained by the way R1-norm is calculated : when the nondominated points of a set *A* are slightly better than those of a set *B* over a large region of the trade-off surface (which seems to be the case here with PAMUC w.r.t. WS), this small advantage is amplified because the outcome function $C(S^{PAMUC},S^{WS},u)$ is equal to 1 when $u^*(S^{PAMUC}) > u^*(S^{WS})$ [see Eq. (4.23)] ; since R1-norm is computed by integrating $C(S^{PAMUC},S^{WS},u)$ over all the utility functions u, the outperformance of PAMUC is exaggerated. This trend is corrected with higher number of points in the trade-off surface.

Problem	**Number of processes**	**Number of runs for each process**	***R1(PAMUC,WS)***	
			Mean	***Standard deviation***
M-UC	*20*	*30*	*0.74*	*0.28*
	1000	*100*	*0.51*	*0.12*

Table 4.16 : Comparison of the weighted sum method and PAMUC for problem M-UC.

4.4.2.2 Test case M-LOC

The second example is due to Kita (cf. [VAN98]) :

$$\max \begin{cases} f_1(\mathbf{x}) = -x_1^2 + x_2 & (4.115) \\ f_2(\mathbf{x}) = x_1/2 + x_2 + 1 & (4.116) \end{cases}$$

$$\text{subject to : } g_1(\mathbf{x}) = x_1/6 + x_2 - 13/2 \leq 0, \quad (4.117)$$

$$g_2(\mathbf{x}) = x_1/2 + x_2 - 15/2 \leq 0, \quad (4.118)$$

$$g_3(\mathbf{x}) = 5\,x_1 + x_2 - 30 \leq 0, \quad (4.119)$$

$$\text{with } 0 \leq x_1 \leq 100 \text{ and } 0 \leq x_2 \leq 100. \quad (4.120)$$

The EA used here is the same as in test case S-CRES (cf. § 3.4.1.4) except that $N = 40$. The process was launched 20 times. Each process consists in running the EA with both methods (PAMUC and WS) 10 times, with a set of weights varying from $\{w_1^* = 0 ; w_2^* = 1\}$ for the first run to $\{w_1^* = 1 ; w_2^* = 0\}$ for the last run, by a constant step. Fig. 4.29 illustrates nondominated solutions obtained by both methods for one process.

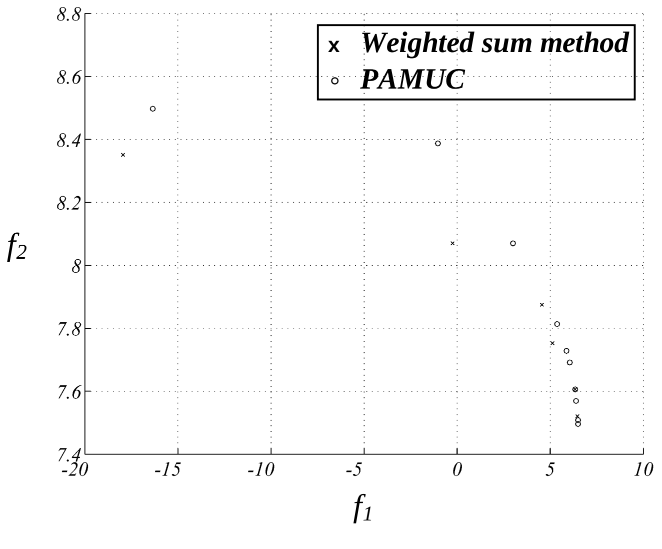


Fig. 4.29 : Nondominated solutions obtained with the weighted sum and PAMUC methods for M-LOC (for one process).

Numerical results are mentioned in Table 4.17, and show that PAMUC outperforms the weighted sum method, although none of them is able to cover efficiently the left side of the PAreto front (i.e. the region such that $-20 < f_1(\mathbf{x}) < 0$).

Problem	*Number of processes*	*R1(PAMUC,WS)*	
		Mean	*Standard deviation*
M-LOC	*20*	*0.88*	*0.11*
	100	*0.86*	*0.09*

Table 4.17 : Comparison of the weighted sum method and PAMUC for problem M-LOC.

Those results are confirmed when a higher number of processes is used. All runs (launched with both methods) furnished feasible solutions.

4.4.2.3 Test case M-LOQC

The third test case is due to Osyczka (cited in [VAN98]), and is formulated as follows :

$$\max \begin{cases} f_1(\mathbf{x}) = x_1 + x_2^{\,2} & (4.121) \\ f_2(\mathbf{x}) = x_1^{\,2} + x_2 & (4.122) \end{cases}$$

$$\text{subject to : } g_1(\mathbf{x}) = 12 - x_1 - x_2 \geq 0, \quad (4.123)$$
$$g_2(\mathbf{x}) = x_1^{\,2} + 10\,x_1 - x_2^{\,2} + 16\,x_2 - 80 \geq 0, \quad (4.124)$$
$$2 \leq x_1 \leq 7 \text{ and } 5 \leq x_2 \leq 10. \quad (4.125)$$

The EA used here is the same as in the previous example (with the same parameters). The process was launched 20 times, each process consisting in running the EA with PAMUC and the weighted sum method 10 times, with a set of weights varying from $\{w_1^* = 0 \; ; w_2^* = 1\}$ for the first run to $\{w_1^* = 1 \; ; w_2^* = 0\}$ for the last run, by a constant step. All solutions (found by both methods) were feasible.

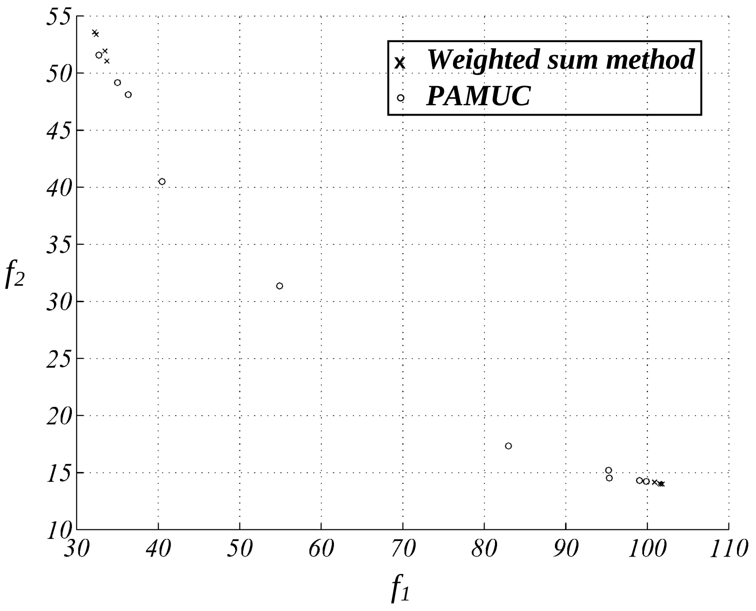


Fig. 4.30 : Nondominated solutions obtained with the weighted sum and PAMUC methods for M-LOQC (for one process).

Problem	**Number of processes**	***R1(PAMUC,WS)***	
		Mean	**Standard deviation**
M-LOQC	*20*	*0.93*	*0.05*

Table 4.18 : Comparison of the weighted sum method and PAMUC for problem M-LOQC.

Statistics on the 20 processes are gathered in Table 4.18, showing that PAMUC gives better results than the weighted sum method. One can also observe in Fig. 4.30 that the points obtained by the weighted sum method are concentrated at the extreme sides of the Pareto front, while the solutions of PAMUC are better distributed along the Pareto front.

At first sight, this result may seem astonishing. Indeed, Jin *et al.* showed in [JIN01] that most aggregation methods were not capable to represent concave Pareto fronts (as the concave PF of this 2-objective maximization problem). Their explanation is based on the following statement : the ability to converge towards a Pareto solution is related to its stability with respect to a given combination of weights. This concept is illustrated in Fig. 4.31 for a 2-objective minimization problem characterized by a convex trade-off surface front.

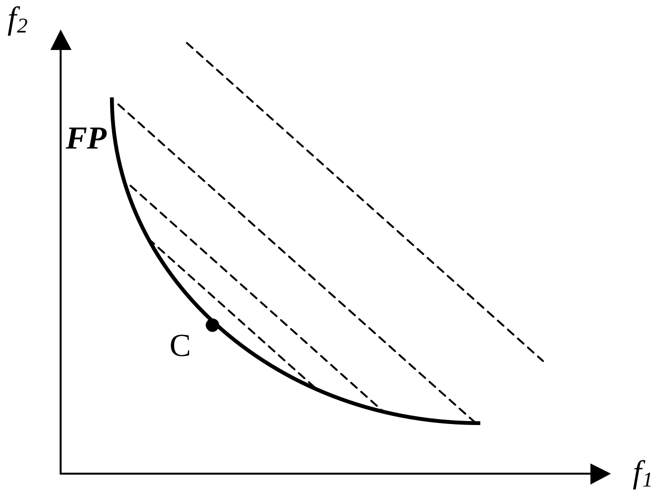


Fig. 4.31 : Convex Pareto front (in a minimization problem).

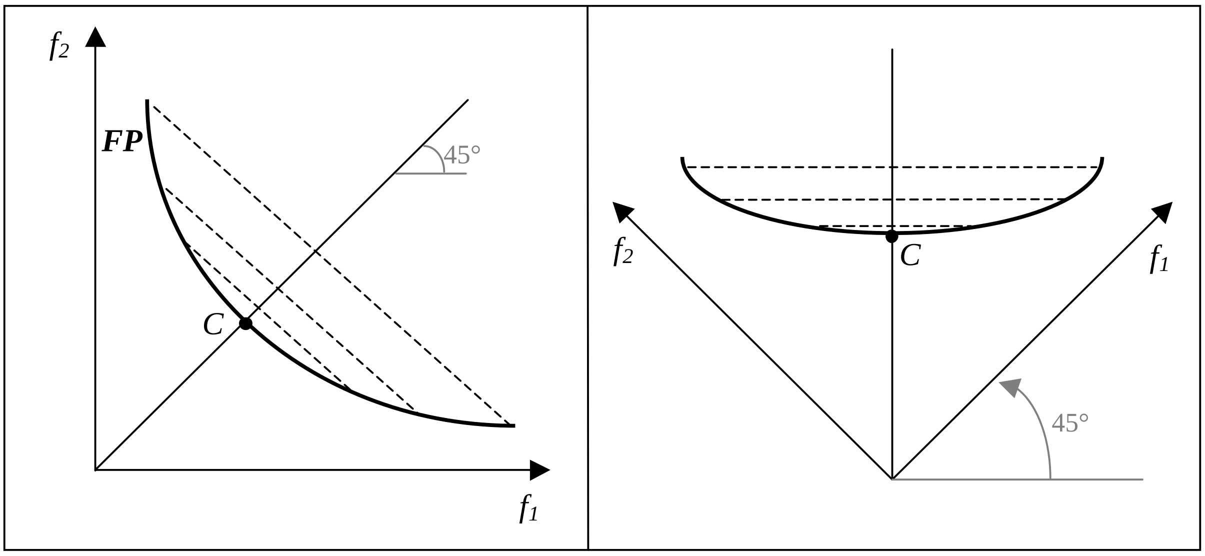


Fig. 4.32 : Seeking the minimum of $w_1 f_1 + w_2 f_2$ with $w_1 = w_2 = 0.5$ (in a minimization problem).

Each network of parallel lines is associated with a slope corresponding to a set of weights $\{w_1, w_2\}$. If $w_1 = w_2 = 0.5$ for instance, a 45° rotation of axes f_1 and f_2 transforms the multiobjective problem into a single-objective one (cf. Fig. 4.32).

With the weighted sum method for example, when the Pareto front is convex, all weight combinations lead to stable nondominated solutions. On the contrary, when it is concave, only points located on extreme sides of the Pareto front (*A* and *B* in Fig. 4.33 ; see also Fig. 4.30 for test case M-LOQC) can be reached.

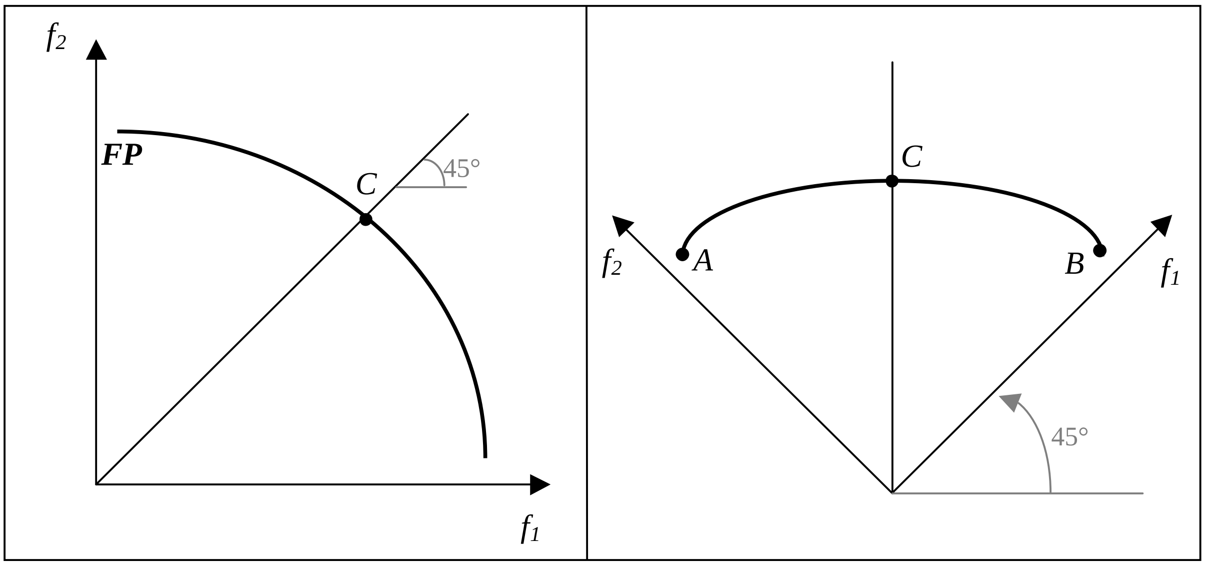


Fig. 4.33 : Seeking the minimum of $\{f_1\ \ f_2\}^T$ with $w_1 = w_2 = 0.5$ with a concave Pareto front (in a minimization problem).

Why has PAMUC found points scattered along the trade-off surface ? To answer this question, a careful examination of PROMETHEE II has to be done. Eq. (3.40) giving the preference flux $\phi(a)$ of an individual *a* (which acts as the fitness function of the EA in PAMUC : see § 3.6.2) can be developed as follows :

$$\phi(a) = \sum_b [\pi(a,b) - \pi(b,a)] = \sum_b [\sum_i w_i P_i(a,b) - \sum_i w_i P_i(b,a)] \qquad (4.126)$$

$$\Leftrightarrow \phi(a) = \sum_i w_i \underbrace{[\sum_b P_i(a,b) - \sum_b P_i(b,a)]}_{\phi_i} \qquad (4.127)$$

where ϕ_i are single-criterion fluxes [MAR89]. When there are no constraints, applying PAMUC is equivalent to perform a weighted sum *not* directly on the objectives, but on the single-criterion fluxes ϕ_i. Here lies the difference with classical linear aggregation methods : instead of a linear combination of the objectives, which is done independently for each individual of the EA, the preference flux ϕ takes the other solutions of the population into account, making the procedure *nonlinear*. Theoretical results valid for linear aggregation techniques are therefore unapplicable : the intrinsic nonlinearity of PAMUC allows it to find solutions even if the Pareto front is concave, though it should be emphasized that a uniform distribution along the Pareto front is not guaranteed for *every* problem characterized by a concave trade-off surface. This matter will be discussed below.

4.4.2.4 Test case M-QOC

The fourth multiobjective problem is due to Srinivas and Deb [DEB01], and is defined as follows :

$$max \begin{cases} f_1(\mathbf{x}) = -[2 + (x_1 - 2)^2 + (x_2 - 1)^2] & (4.128) \\ f_2(\mathbf{x}) = -[9\, x_1 - (x_2 - 1)^2] & (4.129) \end{cases}$$

$$\text{subject to : } g_1(\mathbf{x}) = {x_1}^2 + {x_2}^2 - 225 \le 0, \quad (4.130)$$

$$g_2(\mathbf{x}) = x_1 - 3\, x_2 + 10 \le 0, \quad (4.131)$$

$$-20 \le x_1 \le 20 \text{ and } -20 \le x_2 \le 20. \quad (4.132)$$

Figure 4.34 depicts 10,000 points of the search space randomly created (only feasible points, i.e. satisfying $g_1(x) \le 0$ and $g_2(x) \le 0$, are represented).

The EA used here is the same as in test case S-CRES (cf. § 3.4.1.4) save that $N = 100$. The process was launched 20 times. Each process consists in running the EA with both methods (PAMUC and WS) 10 times, with a set of weights varying from $\{w_1^* = 0 ; w_2^* = 1\}$ for the first run to $\{w_1^* = 1 ; w_2^* = 0\}$ for the last run, by a constant step. Statistics on the 20 processes are mentioned in Table 4.19.

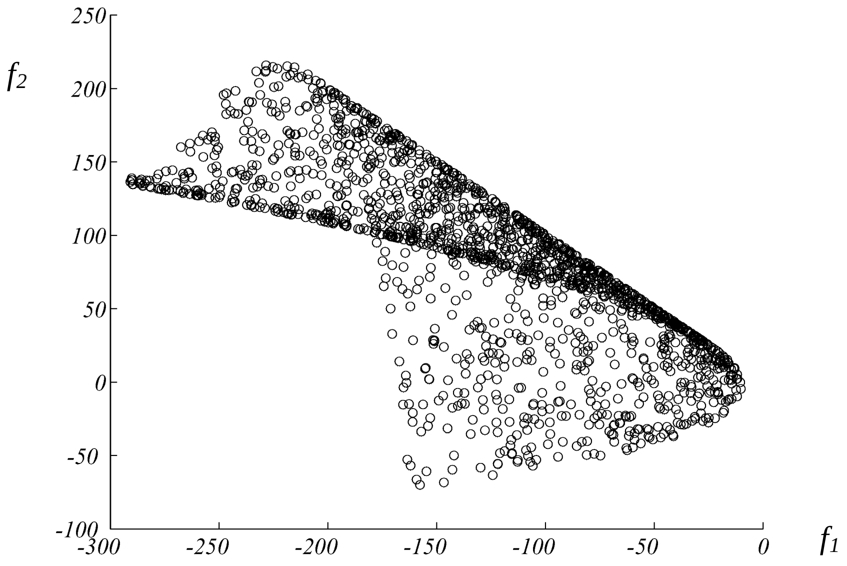


Fig. 4.34 : Feasible solutions (amidst 10,000 points randomly generated) in the objective space for M-QOC.

Figure 4.35 shows nondominated solutions obtained by both methods for one process. While WS results are gathered in a very narrow zone close to (0 ; 0), the solutions obtained by PAMUC cover larger zones at the extreme sides of the Pareto front. Nervertheless, the central part of the trade-off surface is not reached by the EA, whatever combination of weight is used. This illustrates the warning made in previous section, hence a limitation of PAMUC.

Problem	*Number of processes*	*R1(PAMUC,WS)*	
		Mean	*Standard deviation*
M-QOC	*20*	*0.97*	*0.09*

Table 4.19 : Comparison of the weighted sum method and PAMUC for problem M-QOC.

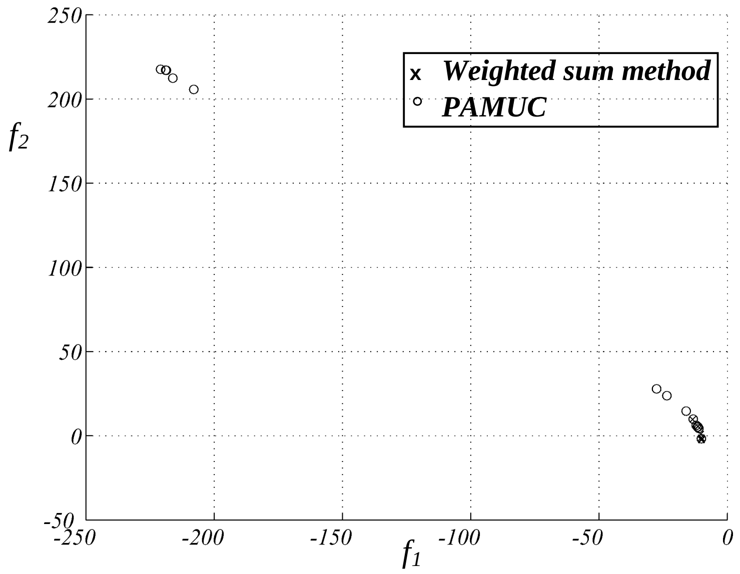


Fig. 4.35 : Nondominated solutions obtained with the weighted sum and PAMUC methods for M-QOC (for one process).

All runs made with PAMUC gave feasible solutions, while 91% of runs made with Joines Houck's method gave birth to solutions satisfying all the constraints. Figure 4.36 shows the rate of feasible individuals with respect to the generations for one run of the EA (with $w_1^* = w_2^* = 0.5$) ; once again PAMUC enables a faster convergence towards the feasible domain, whilst the dynamic penalty method has some hindrances to keep enough admissible individuals during the generations.

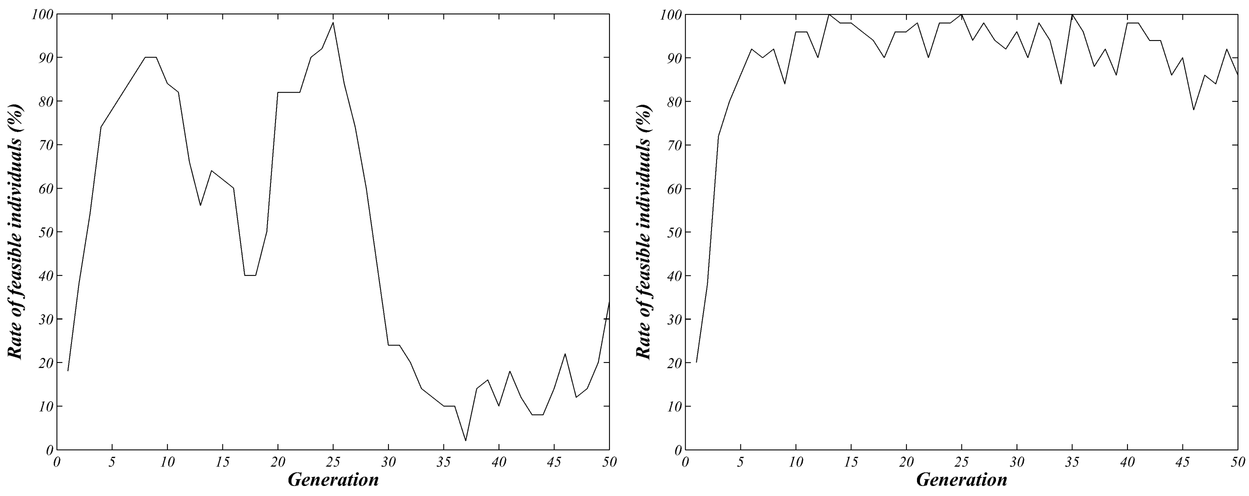


Fig. 4.36 Rate of feasible individuals w.r.t. the generation for M-QOC (with weighted sum [left] and PAMUC [right] methods).

4.4.2.5 Test case M-DPF

As already underlined before (cf. § 3.5.2), there are rather few studies specially devoted to constrained *and* multiobjective evolutionary optimization, though it is encountered very commonly in industrial applications.

In order to validate techniques dealing with constrained optimization with multiple criteria in EAs, an important breakthrough was brought by Deb [DEB99,DEB01], who developed tunable test cases, from which examples M-DPF and M-LFS were taken.

The test generator is defined as follows :

$$\min \begin{cases} f_1(\mathbf{x}) = x_1 & (4.133) \\ f_2(\mathbf{x}) = g(\mathbf{x})\,[1 - f_1(\mathbf{x})/g(\mathbf{x})] & (4.134) \end{cases}$$

$$\text{subject to : } c(\mathbf{x}) = \cos(\theta)\,[\,f_2(\mathbf{x}) - e] - \sin(\theta)\,f_1(\mathbf{x}) \geq a\,|\,\sin(b\pi(\sin(\theta)(f_2(\mathbf{x}) - e) + \cos(\theta)f_1(\mathbf{x}))^c)|^d \qquad (4.135)$$

$$0 \leq x_i \leq 1 \quad \text{for } i = 1,\ldots, 5, \qquad (4.136)$$

$$\text{with : } g(\mathbf{x}) = x_1^{\,2} + x_2^{\,2} + x_3^{\,2} + x_4^{\,2} + x_5^{\,2}, \qquad (4.137)$$

where *g(x)* can actually be any multimodal function. Different levels of difficulty can be obtained by changing *g(x)*. De Jong's function with 5 variables [WHI96] was chosen.

The following parameters were used : $\theta = -\,0.2\pi$, $a = 0.2$, $b = 3$, $c = 1$, $d = 6$ and $e = 1$. The EA used here is the same as in the previous example (with the same parameters, except that $N_{gen} = 100$). The process was launched 20 times. Each process consists in running the EA with PAMUC and the weighted sum method 10 times, with a set of weights varying from $\{w_1^* = 0 \,;\, w_2^* = 1\}$ for the first run to $\{w_1^* = 1 \,;\, w_2^* = 0\}$ for the last run, by a constant step. Statistics on the 20 processes show that the mean value for *R1(PAMUC,WS)* is equal to 0.86 (see Table 4.20). All solutions (found by both methods) were feasible.

In Fig. 4.37, a set of feasible solutions (among 10,000 objective vectors randomly generated) illustrates the fact that the Pareto front is discontinuous, and that solutions found by PAMUC are closer to the Pareto front, in comparison with the weighted sum method. It should be noted that both methods found objective vectors on each of the four parts composing the discontinuous trade-off surface.

Problem	***Number of processes***	***R1(PAMUC,WS)***	
		Mean	***Standard deviation***
M-DPF	*20*	*0.86*	*0.13*
	100	*0.83*	*0.08*

Table 4.20 : Comparison of the weighted sum method and PAMUC for problem M-DPF.

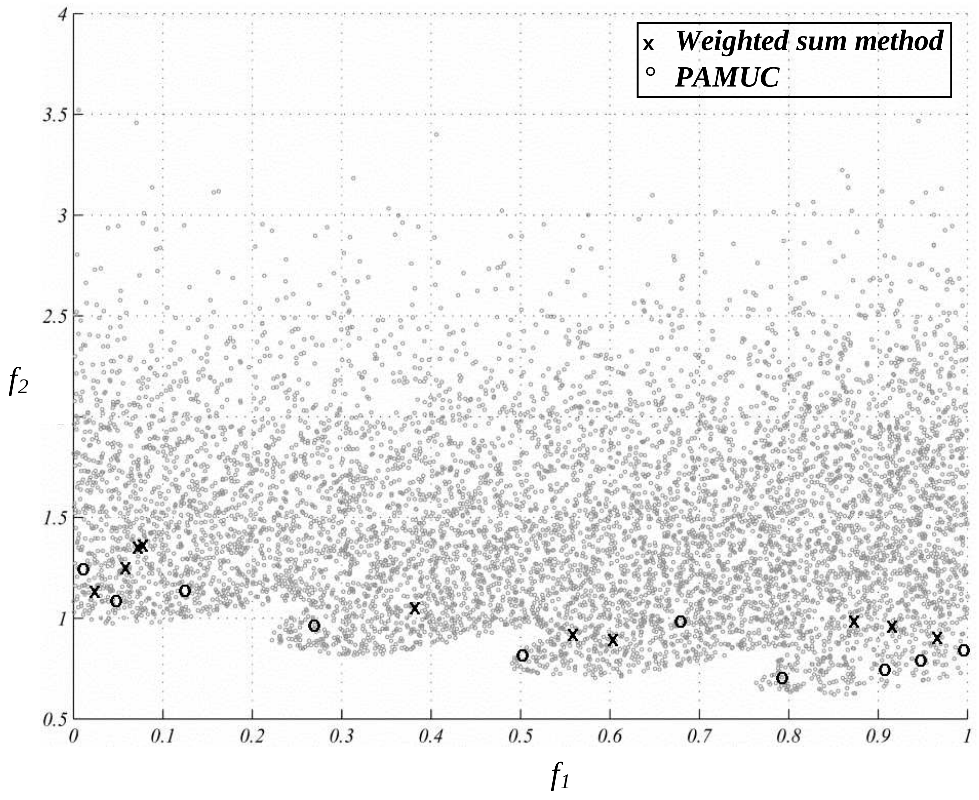


Fig. 4.37 : Nondominated solutions obtained with the weighted sum and PAMUC methods for M-DPF (for one process).

4.4.2.6 Test case M-LFS

The functions, constraints, variables and parameters of the study are the same as in the previous example, save that $\theta = 0.1\pi$, $a = 40$, $b = 4$, $c = 1$, $d = 2$ and $e = -1$. The process was launched 20 times. Each process consists in running the EA with PAMUC and the weighted sum method 10 times, with a set of weights varying from $\{w_1^* = 0 ; w_2^* = 1\}$ for the first run to $\{w_1^* = 1 ; w_2^* = 0\}$ for the last run, by a constant step.

Figure 4.38 shows a set of admissible solutions (among 10,000 objective vectors randomly created), illustrating that the feasible domain is a discontinuous space composed of 8 layers. The nondominated solutions are located on layer L_1. Statistics on the 20 processes are collected in Table 4.21. Once again it shows a better distribution of the points along the trade-off surface, as well as a better behaviour in terms of handling of the constraints.

Parameters of the study	*Mean*	*Standard Deviation*
R1(PAMUC,WS) norm	0.82	0.14
Rate of feasible individuals per process (PAMUC)	0.96	0.18
Rate of feasible individuals per process (WS)	0.76	0.16
Rate of feasible ind. located on layer L_1 (PAMUC) *	0.65	0.12
Rate of feasible ind. located on layer L_1 (WS) *	0.31	0.09

Table 4.21 : Results for M-LFS

(: ratio between the total number of feasible solutions and the number of feasible solutions on layer L1).*

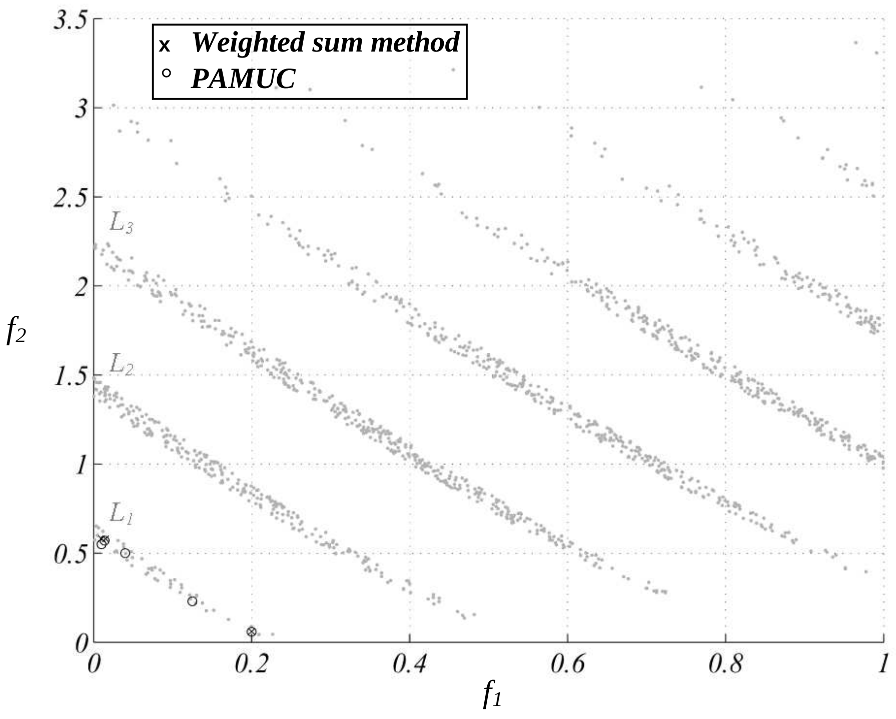


Fig. 4.38 : Nondominated solutions obtained with the weighted sum and PAMUC methods for M-LFS (for one process).

Computations have been made for test cases M-DPF and M-LFS by using a more challenging function $g(\mathbf{x})$, namely Rastrigin's function (instead of De Jong's function) :

$$g(\mathbf{x}) = \sum_{i=1}^{5} [1+ x_i^2 - \cos(2\pi x_i)]. \tag{4.138}$$

Numerical results show that PAMUC and the weighted sum method both have difficulties whilst endeavouring to find solutions close to the Pareto front : when feasible objective vectors are found, they are generally far from it. In those cases, an *a posteriori* method should be used to determine the trade-off surface, and a multicriteria decision aid method applied only at the end of the search process.

4.4.2.7 Test case M-BDP

The 7th multiobjective test case is a mechanical benchmark : the beam design problem (cf. Fig. 4.39), defined in [OSY02] :

$$\min \begin{cases} f_1(\mathbf{x}) = b.l. \sum_{n=1}^{6} x_n & (4.139) \\ \\ f_2(\mathbf{x}) = \dfrac{Fl^3}{2E}\left(\dfrac{1}{I_1} + \sum_{n=2}^{6} \dfrac{n^3-(n-1)^3}{I_n}\right) & (4.140) \end{cases}$$

where $f_1(\mathbf{x})$ is the volume of the beam and $f_2(\mathbf{x})$ is the displacement under the force F and I_n is the moment of inertia for each part of the beam :

$$I_n = \frac{bx_n^3}{12} \quad \text{for } n = 1,\ldots, 6. \tag{4.141}$$

The variables x_n are the thicknesses of the beam components. It is a discrete programming problem, since :

$$X_n \in \{12,14,16,18,20,22,24,26,28,30,32\}\ [mm]\ \text{for } n = 1,\ldots, 6. \tag{4.142}$$

The first 6 constraints express that the normal stress must lie under the maximum authorized level σ_g = 360 N/mm^2, and the last ones impose a linear restriction on the thicknesses :

$$c_n = \frac{6Fnl}{bx_n^2} \le \sigma_g \qquad \text{for } n = 1,\ldots, 6, \tag{4.143}$$

$$c_{n+6} = x_{n+1} - x_n \ge 0 \quad \text{for } n = 1,\ldots, 5. \tag{4.144}$$

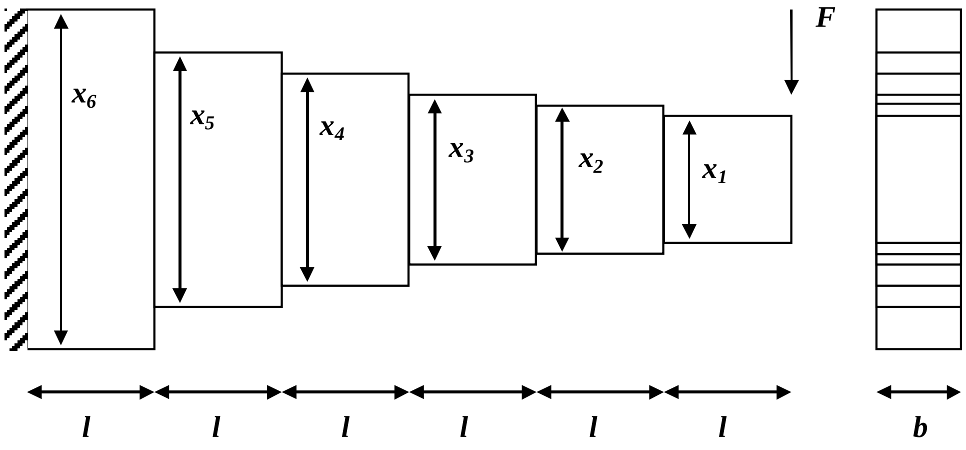


Fig. 4.39 : Osyczka beam design problem.

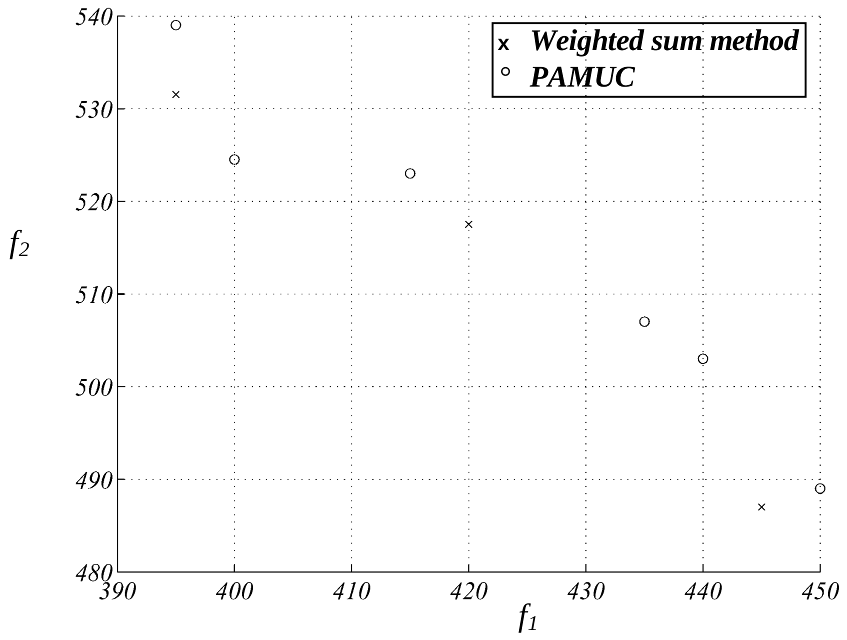


Fig. 4.40 : Nondominated solutions obtained with the weighted sum and PAMUC methods for M-BDP (for one process).

Table 4.22 gathered the EA parameters used for the study.

Symbol	*Parameter*	*Value*
Env	*Environment*	*Std-EA Matlab*
Cod	*Coding of the variables*	*Gray*
N	*Size of the population*	*50*
N_{gen}	*Number of generations*	*50*
T_s	*Type of selection*	*Tournament*
n_t	*Number of individuals participating to a tournament*	*5*
p_c	*Probability of crossover*	*0.8*
T_c	*Type of crossover*	*2-point*
p_m	*Probability of mutation*	*0.05*
T_m	*Type of mutation*	*Flip*

Table 4.22 : EA parameters for test case M-BDP.

The process was launched 20 times. Each process consists in running the EA with both methods (PAMUC and WS) 10 times, with a set of weights varying from $\{w_1^* = 0 ; w_2^* = 1\}$ for the first run to $\{w_1^* = 1 ; w_2^* = 0\}$ for the last run, by a constant step. Statistics on the 20 processes are gathered in Table 4.23. Figure 4.40 shows nondominated solutions obtained by both methods for one process. All results (whatever method used) produced feasible solutions.

Problem	*Number of processes*	*R1(PAMUC,WS)*	
		Mean	*Standard deviation*
M-BDP	*20*	*0.79*	*0.18*

Table 4.23 : Comparison of the weighted sum method and PAMUC for problem M-BDP.

4.4.2.8 Test case M-3OU

So far only 2-objective optimization have been investigated. In fact, as Deb pointed out in [DEB02], many multiobjective algorithms are validated only on test cases characterized by two criteria. However, of course, real-life applications may contain more than 2 objectives.

Therefore, the 8th and 9th multiobjective problems will test examples with 3 objectives. They are due to Viennet (cited in [COE02a]), and the first one is formulated as follows :

$$\max \begin{cases} f_1(\mathbf{x}) = x_1^2 + (x_2 - 1)^2 & (4.145) \\ f_2(\mathbf{x}) = x_1^2 + (x_2 + 1)^2 + 1 & (4.146) \\ f_3(\mathbf{x}) = (x_1 - 1)^2 + x_2^2 + 2 & (4.147) \end{cases}$$

$$\text{with :} \quad -2 \le x_1 \le 2 \ \text{ and } \ -2 \le x_2 \le 2. \tag{4.148}$$

Figure 4.41 illustrates a set of 10,000 objective vectors randomly generated drawn in the objective space.

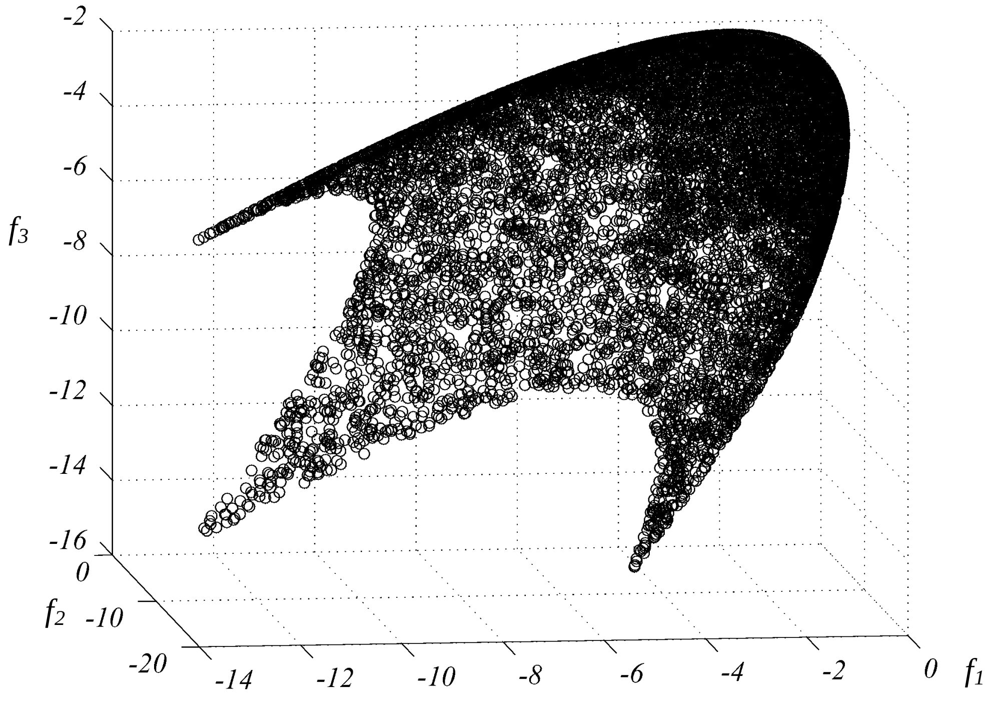


Fig. 4.41: Random population of 10,000 individuals in the objective space for M-3OU.

The process was launched 10 times. Each process consists in running the EA with both methods (PAMUC and WS) 210 times, with varying weights as explained in § 4.3.5. Statistics on the 10 processes are shown in Table 4.24.

Problem	***Number of processes***	***R1(PAMUC,WS)***	
		Mean	***Standard deviation***
M-3OU	*10*	*0.60*	*0.23*

Table 4.24 : Comparison of the weighted sum method and PAMUC for problem M-3OU.

4.4.2.9 Test case M-3OC

The second 3-objective problem is also due to Viennet (cf. [COE02a]), and has three constraints :

$$\max \begin{cases} f_1(\mathbf{x}) = (x_1 - 2)^2/2 + (x_2 + 1)^2/13 + 3 & (4.149) \\ f_2(\mathbf{x}) = (x_1 + x_2 - 3)^2/175 + (2\,x_2 - x_1)^2/17 - 13 & (4.150) \\ f_3(\mathbf{x}) = (3\,x_1 - 2\,x_2 + 4)^2/8 + (x_1 - x_2 + 1)^2/27 + 15 & (4.151) \end{cases}$$

$$\text{subject to : } g_1(\mathbf{x}) = -4\,x_1 - x_2 + 4 > 0, \qquad (4.152)$$
$$g_2(\mathbf{x}) = x_1 + 1 > 0, \qquad (4.153)$$
$$g_3(\mathbf{x}) = x_2 - x_1 + 2 > 0, \qquad (4.154)$$

$$\text{with : } -4 \leq x_1 \leq 4 \text{ and } -4 \leq x_2 \leq 4. \qquad (4.155)$$

Figure 4.42 illustrates a set of objective vectors (among 10,000 points randomly generated) drawn in the objective space.

The process was launched 10 times. Each process consists in running the EA with both methods (PAMUC and WS) 210 times, with different weights as explained in § 4.3.5. Statistics on the 10 processes are mentioned in Table 4.25.

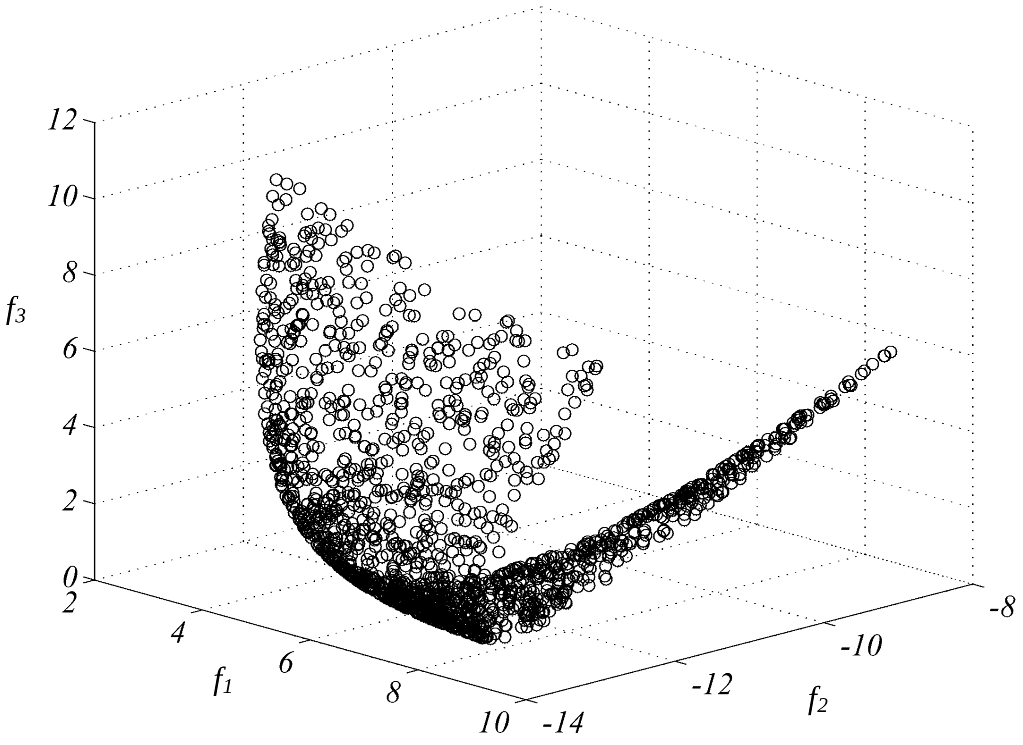


Fig. 4.42 : Feasible population (among 10,000 individuals randomly generated) in the objective space for M-3OC.

Problem	**Number of processes**	**R1(PAMUC,WS)**	
		Mean	**Standard deviation**
M-3OC	10	0.80	0.29

Table 4.25 : Comparison of the weighted sum method and PAMUC for problem M-3OC.

A summary of the numerical results will take place at § 4.4.3, and conclusions will be drawn about the pertinency of using PAMUC to solve multiobjective contrained problems in design optimization. Afterwards, three important computational aspects still have to be considered : the influence of PAMUC parameters p_i and q_i on the results (§ 4.4.4), the sensitivity of the solutions with respect to the user's weights w_i^* (§ 4.4.5) and the computational time (§ 4.4.6).

4.4.3 Summary of multiobjective problems

The multiobjective test cases presented above were aimed to show that PAMUC can reflect correctly the user's preferences. In order to demonstrate PAMUC efficiency for various situations, they had to present different peculiarities (as prescribed by Deb in [DEB99]) ;

- *convexity* of the trade-off surface (e.g. M-UC, M-LOC) ;
- *concavity* of the trade-off surface (e.g. M-LOQC) ;
- *discontinuity* of the trade-off surface (e.g. M-DPF) ;
- presence of *"local" Pareto fronts*, i.e. dominated regions which attract the individuals of the EA (cf. M-LFS) ;
- *n-dimensional objective spaces* with $n \geq 3$ (e.g. M-3OU, M-3OC).

Table 4.26 recapitulates the comparative results between the weighted sum method and PAMUC for the multiobjective test cases. PAMUC clearly outperforms the weighted sum method.

R1(PAMUC,WS)	***M-UC***	***M-LOC***	***M-LOQC***	***M-QOC***	***M-DPF***	***M-LFS***	***M-BDP***	***M-3OU***	***M-3OC***
Mean	*0.74*	*0.88*	*0.93*	*0.97*	*0.86*	*0.82*	*0.79*	*0.60*	*0.80*
Std. deviation	*0.08*	*0.11*	*0.05*	*0.09*	*0.13*	*0.14*	*0.18*	*0.23*	*0.29*

Table 4.26 : Comparison of PAMUC and WS methods for 9 multiobjective methods :
values of the mean and standard deviation of R1-norm.

Jin *et al.* explained in [JIN00] why linear aggregation methods are not capable of finding other solutions than points located at the extreme sides of the Pareto front. However, PROMETHEE II – hence PAMUC – does not imply a pure linear addition of the criteria, since preference functions have to be computed first, comparing each individual to the rest of the population. This explains why objective vectors can be found along the trade-off surface, even when it is concave. The other a priori methods, as the goal programming or the min-max method (cf. § 3.3.3), have not been compared to PAMUC since they demand a different kind of information from the user : instead of (or in addition to) weights, they require the user to define a specific point in the objective space (e.g. the target in goal programming methods).

It is important to emphasize that no claim is done that PAMUC is able to cover the whole Pareto front in all cases, i.e. whatever shape it may have (cf. M-DPF and M-LFS with *g(x)* equal to Rastrigin's function). In very hard problems, an a posteriori method should thus be used first (e.g. NSGA-II [DEB02a] or NPGA2 [ERI01]), and a solution chosen among the nondominated points after the search process.

Before drawing the general conclusions about PAMUC, the following paragraphs will focus on three aspects : the influence of parameters p_i and q_i on the results, the sensitivity of the weights and the computational time.

4.4.4 Influence of parameters p_i and q_i

As exposed in the description of PAMUC, two additional parameters p_i and q_i are needed to run the algorithm (for each criterion). In industrial applications, they can be determined by the decision maker, with respect to his/her knowledge about the objectives, but when such information is not available – which is the case in the mathematical examples presented above –, values of $p_i = 1$ and $q_i = 0$ have been used systematically, and provided good results. In order to validate this choice, example M-UC will be treated with different values of p_i and q_i to investigate their influence on the results.

p_i	$w_1^* = 0$		$w_1^* = 0.25$		$w_1^* = 0.5$		$w_1^* = 0.75$		$w_1^* = 1$	
	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}
0.2	– 0.4075	0.9825	0.3088	0.7108	0.5237	0.4825	0.6973	0.3266	0.9627	– 0.4128
0.5	– 0.3715	0.9886	0.3186	0.6899	0.5039	0.5082	0.6947	0.2937	0.9889	– 0.3753
1	– 0.3718	0.9762	0.3427	0.6746	0.5856	0.4167	0.7092	0.3225	0.9644	– 0.3872
2	– 0.4003	0.9892	0.2785	0.7344	0.5194	0.5211	0.6952	0.3118	0.9978	– 0.4203
5	– 0.3664	0.9364	0.3018	0.6809	0.5355	0.5123	0.6732	0.3198	0.9751	– 0.3819
10	– 0.4020	0.9666	0.3081	0.7238	0.5276	0.5265	0.6650	0.2377	0.9740	– 0.3990

Table 4.27 : Analysis of the influence of parameter ***p_i*** *(with $q_i = 0$) on results for example M-UC,*
with different values of the weights – each number represents the ***mean*** *on 50 runs of the algorithm.*

p_i	$w_1^* = 0$		$w_1^* = 0.25$		$w_1^* = 0.5$		$w_1^* = 0.75$		$w_1^* = 1$	
	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}
0.2	0.1296	0.1182	0.1992	0.1512	0.1681	0.2820	0.1866	0.0987	0.2259	0.1503
0.5	0.1438	0.0738	0.2136	0.2124	0.2473	0.1950	0.2314	0.1712	0.0958	0.1359
1	0.2123	0.1117	0.1310	0.2730	0.1141	0.3545	0.1537	0.1503	0.1284	0.2340
2	0.1367	0.0503	0.2051	0.1301	0.2179	0.1665	0.2018	0.1042	0.0140	0.0535
5	0.2293	0.2831	0.2624	0.2774	0.0366	0.2200	0.2746	0.1335	0.1126	0.2173
10	0.1458	0.1814	0.1534	0.0944	0.1486	0.1878	0.2963	0.3004	0.1294	0.1942

Table 4.28 : Analysis of the influence of parameter ***p_i*** *(with $q_i = 0$) on results for example M-UC,*
with different values of the weights – each number represents the ***standard deviation*** *on 50 runs of the algorithm.*

q_i	$w_1^* = 0$		$w_1^* = 0.25$		$w_1^* = 0.5$		$w_1^* = 0.75$		$w_1^* = 1$	
	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}
0	– 0.4187	0.9995	0.3034	0.6863	0.4747	0.5512	0.6577	0.2838	0.9315	– 0.4143
10^{-5}	– 0.3987	0.9808	0.3134	0.7078	0.4970	0.5188	0.7260	0.3230	0.9496	– 0.4074
10^{-4}	– 0.4206	0.9845	0.2953	0.7157	0.5173	0.5218	0.6704	0.2918	0.9874	– 0.4071
10^{-3}	– 0.4005	0.9869	0.3116	0.6958	0.5405	0.5397	0.7216	0.2944	0.9398	– 0.3945
10^{-2}	– 0.4129	0.9985	0.2931	0.7289	0.5840	0.4430	0.6702	0.3273	0.9980	– 0.4243
10^{-1}	– 0.4249	0.9842	0.2981	0.7339	0.5304	0.4944	0.7177	0.3287	0.8825	– 0.3577

Table 4.29 : Analysis of the influence of parameter ***q_i*** *(with $p_i = 1$) on results for example M-UC,*
with different values of the weights – each number represents the ***mean*** *on 50 runs of the algorithm.*

q_i	$w_1^* = 0$		$w_1^* = 0.25$		$w_1^* = 0.5$		$w_1^* = 0.75$		$w_1^* = 1$	
	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}
0	0.0275	0.0034	0.2252	0.2818	0.2751	0.1036	0.3031	0.2194	0.3069	0.1950
10^{-5}	0.1864	0.1262	0.1775	0.1734	0.2267	0.2083	0.0096	0.0131	0.1994	0.2089
10^{-4}	0.1172	0.0834	0.1908	0.1900	0.1969	0.0969	0.2871	0.1423	0.0682	0.1497
10^{-3}	0.1494	0.0528	0.1097	0.1937	0.0241	0.0243	0.0515	0.1964	0.2142	0.2821
10^{-2}	0.0490	0.0069	0.1931	0.0921	0.1125	0.2951	0.2655	0.1343	0.0071	0.0540
10^{-1}	0.0924	0.1030	0.1441	0.0796	0.1699	0.2184	0.0633	0.0911	0.4157	0.2807

Table 4.30 : Analysis of the influence of parameter ***q_i*** *(with $p_i = 1$) on results for example M-UC,*
with different values of the weights – each number represents the ***standard deviation*** *on 50 runs of the algorithm.*

The parameters of the EA are the same as in § 4.4.2.1. Tables 4.27 to 4.30 show the results of PAMUC with different values of the user's weights and of p_i (resp. q_i). In all cases, $p_1 = p_2$, $q_1 = q_2$, and of course $w_2^* = 1 - w_1^*$. The values of p_i are greater than (or equal to) 0.2, and the values q_i remain less than (or equal to) 0.1.

The numerical results clearly indicate that whichever values of p_i or q_i are used (within the given range), the solutions are within the range defined by the standard deviation. Therefore, when no additional information is known about the criteria, values of $p_i = 1$ and $q_i = 0$ are recommended.

4.4.5 Sensitivity analysis with respect to the weights w_i^*

Another aspect is related to the sensitivity of the weights. It is well known that EAs are robust against noise, but in the particular case of PAMUC, it is important to check whether a slight modification of the set of weights would induce an important variation of the results. Therefore, a sensitivity analysis on the weights w_i^* was performed, by comparing results obtained for 5 different values of the weights ($w_1^* \in \{ 0 ; 0.25 ; 0.5 ; 0.75 ; 1\}$) with results furnished for corresponding perturbated weights ($w_1^* \in \{ 0.01 ; 0.26 ; 0.51 ; 0.76 ; 0.99\}$). Table 4.31 depicts the mean and standard deviation over 100 runs, with the same EA parameters as above.

Though a more in-depth sensitivity analysis can be done (cf. approach proposed in [WOL95] for instance), this is meaningless here since a quick examination of Table 4.31 shows that the standard deviation of the solutions and the variations of the results have the same range. It can thus be concluded that solutions are not sensitive to small variations of the weights.

		$w_1^* = 0$		$w_1^* = 0.25$		$w_1^* = 0.5$		$w_1^* = 0.75$		$w_1^* = 1$	
		f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}	f_1^{opt}	f_2^{opt}
Mean	w^*	– 0.4075	0.9825	0.3088	0.7108	0.5237	0.4825	0.6973	0.3266	0.9627	– 0.4128
	w^*_{pert}	– 0.3715	0.9886	0.3186	0.6899	0.5039	0.5082	0.6947	0.2937	0.9889	– 0.3753
Std. Dev.	w^*	0.1296	0.1182	0.1992	0.1512	0.1681	0.2820	0.1866	0.0987	0.2259	0.1503
	w^*_{pert}	0.1438	0.0738	0.2136	0.2124	0.2473	0.1950	0.2314	0.1712	0.0958	0.1359

Table 4.31 : Influence of the variation of weights on the solutions for example M-UC (each number represents the mean or the standard deviation over 100 runs).

4.4.6 Algorithmic complexity

This section is devoted to the calculus of PAMUC computational complexity. Its differences with the traditional weighted sum method (WS) – Joines and Houck's penalty method being used to handle the constraints – lie in 3 parts of the algorithm (cf. Fig. 4.43) :

- during the selection scheme :
  - all individuals are compared thanks to PAMUC following m+1 objectives ;
  - additionaly, an elitist selection procedure is applied wherein each 4-member set of 2 parents and their corresponding children are compared by PROMETHEE II ;
- the weights have to be updated at each generation, hence the number of feasible individuals computed.

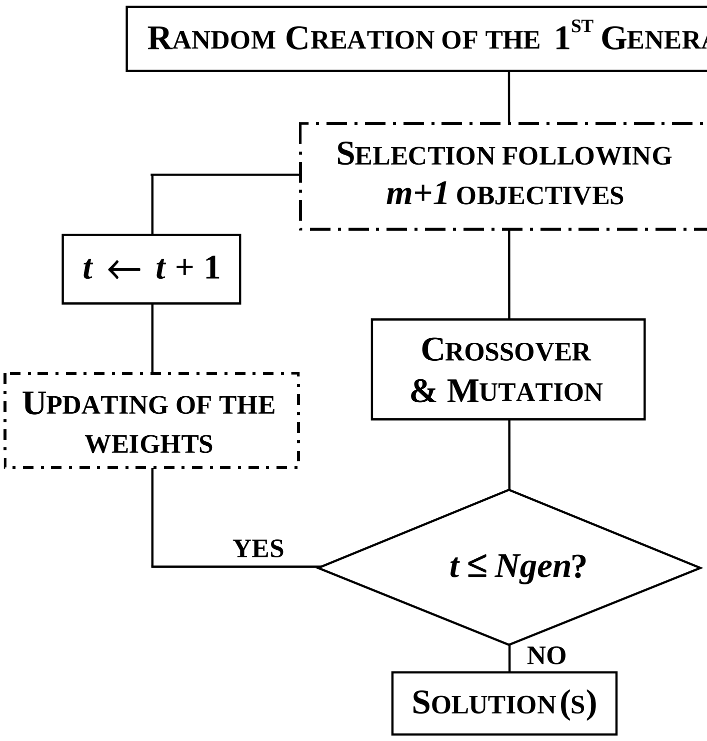

RANDOM CREATION OF THE 1ST GENERATION (t=1)
SELECTION FOLLOWING m+1 OBJECTIVES
t ← t + 1
CROSSOVER & MUTATION
UPDATING OF THE WEIGHTS
YES
t ≤ Ngen?
NO
SOLUTION(S)

Fig. 4.43 : Flow-chart of PAMUC
(the modules having differences with WS are characterized by dash-dotted lines).

A decomposition of the different parts of the standard EA (cf Fig. 4.43) combined with the weighted sum method leads to the following formula :

$$T_{1gen}^{WS} = N.[T_{sel} + T_{cross} + T_{mut} + T_{obj} + T_{constr} + K], \tag{4.156}$$

where :

- T_{1gen}^{WS} is the computational time needed for one generation of the EA with the weighted sum method ;
- N is the size of the population ;
- T_{sel} is the average time (for one member of the population) to perform selection ;
- T_{cross} is the average time (for one member of the population) to perform crossover ;
- T_{mut} is the average time (for one member of the population) to perform mutation ;
- T_{obj} is the time needed to compute the values of the m objective functions (for one individual) ;
- T_{constr} is the time needed to compute the values of the $p+q$ (equality and inequality) constraints (for one individual) ;
- K is a second-order term for remaining (low cost) computations of the algorithm.

For PAMUC, similar developments lead to Eq. (4.157) :

$$\begin{aligned} T_{1gen}^{PAMUC} = \; & N.[T_{sel\_PROM\_II} + T_{cross} + T_{mut} + T_{obj} + T_{constr} + (N+1).(m+1). \\ & T_{PROM\_II} + K'], \end{aligned} \tag{4.157}$$

where :

- T_{1gen}^{PAMUC} is the computational time needed for one generation of the EA with PAMUC ;
- $T_{sel\_PROM\_II}$ is the average time (for one member of the population) to perform selection (by an elitist selection procedure using PROMETHEE II for each pair of parents and their corresponding children) ;
- m is the number of objective functions ;
- T_{PROM_II} is the time needed to compute (for a couple of individuals (a,b)) the preference functions $P_i(a,b)$ and the preference indexes $\pi(a,b)$ needed to rank the individuals in PROMETHEE II ;
- K' is a second-order term for remaining (low cost) computations of the algorithm.

T_{cross} and T_{mut} are functions of the coding and the number of variables. With increasing sizes of population, number of generations and of objectives, the asymptotic complexity AC of the EA are given by the following expressions :

$$AC^{WS} = o(N_{gen} \,.\, N \,.\, m), \tag{4.158}$$

$$AC^{PAMUC} = o(N_{gen} \,.\, N^2 . m), \tag{4.159}$$

where the notation $f = o(g)$ means that :

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 0 \,. \tag{4.160}$$

Though this theoretical result demonstrates that PAMUC is more expensive than a simple linear aggregation of the weights for large computations (with high numbers of generations N_{gen} and members in the population N), it must be underlined that in most cases, in pre-design optimization, EAs are used with $N \leq 200$ and $N_{gen} \leq 1000$, making the differences between both methods very reasonable. The calculations presented below aim to illustrate the fact that the use of PAMUC, besides from giving better results than the weighted sum method, needs only a minor supplementary CPU time to perform it.

Indeed, Tables 4.32 to 4.34 present the CPU times of the EA for 3 emblematic test cases (in which N and N_{gen} are variable whereas the other EA parameters are the same as in §§ 4.4.1 and 4.4.2) :

- S-38IC (characterized by 5 variables, 1 objective and 38 inequality constraints) ;

- M-BDP (characterized by 6 variables, 2 objectives and 11 inequality constraints, $w_1^* = w_2^* = 0.5$) ;

- M-3OC (characterized by 2 variables, 3 objectives and 3 inequality constraints, $w_1^* = w_2^* = w_3^* = ⅓$).

N	*N_{gen}*	*CPU time (s)*		
		Joines and Houck	***PAMUC***	***Relative difference (%)***
50	*50*	*8.6*	*9.4*	*8.5%*
50	*100*	*17.2*	*18.9*	*8.6%*
50	*1000*	*282.8*	*302.0*	*6.4%*
100	*50*	*16.0*	*17.4*	*8.1%*
100	*100*	*29.1*	*35.0*	*16.8%*
100	*1000*	*529.0*	*567.8*	*6.8%*
200	*50*	*34.6*	*41.5*	*16.5%*
200	*100*	*56.8*	*70.7*	*19.7%*
200	*1000*	*1034.2*	*1190.3*	*13.1%*

Table 4.32 : CPU time for different values of the number of generations N_{gen} and the size of the population N (for test case S-38IC).

N	*N_{gen}*	*CPU time (s)*		
		WS + Joines and Houck	***PAMUC***	***Relative difference (%)***
50	*50*	*4.9*	*6.5*	*24.5%*
50	*100*	*9.7*	*12.7*	*23.4%*
50	*1000*	*184.0*	*185.0*	*0.5%*
100	*50*	*12.9*	*14.2*	*8.6%*
100	*100*	*18.2*	*30.7*	*40.8%*
100	*1000*	*299.6*	*374.2*	*19.9%*
200	*50*	*32.1*	*40.7*	*21.3%*
200	*100*	*35.5*	*61.7*	*42.5%*
200	*1000*	*572.4*	*848.6*	*32.5%*

Table 4.33 : CPU time for different values of the number of generations N_{gen} and the size of the population N (for test case M-BDP).

N	*N_{gen}*	*CPU time (s)*		
		WS + Joines and Houck	***PAMUC***	***Relative difference (%)***
50	*50*	*6.9*	*7.3*	*5.9%*
50	*100*	*11.0*	*13.6*	*19.1%*
50	*1000*	*179.0*	*194.4*	*7.9%*
100	*50*	*11.7*	*13.6*	*13.9%*
100	*100*	*20.1*	*27.9*	*28.0%*
100	*1000*	*316.5*	*395.4*	*19.9%*
200	*50*	*26.5*	*31.1*	*15.0%*
200	*100*	*38.4*	*63.6*	*39.7%*
200	*1000*	*614.2*	*872.4*	*29.6%*

Table 4.34 : CPU time for different values of the number of generations N_{gen} and the size of the population N (for test case M-3OC).

Those results (obtained on a PC with *freq.* = 900 MHz in MS Windows 2000 environment) corroborate the trend given by the theory, i.e. when the size of the population (all other parameters remaining equal) increases, the relative difference between both methods (in terms of computational cost) also increases ; however, in all test cases analyzed here, even for a rather high number of individuals (N = 200), this gap remains reasonable (less than 50% of supplementary CPU time needed).

4.5 Conclusions

This chapter was devoted to a rigorous validation of PAMUC. The test cases were divided in two categories :

- *single-objective* test cases : they demonstrate that PAMUC, though initially developed for multicriteria optimization, is an efficient tool in comparison with specific constraint-handling techniques. The good results are mainly due to the adaptivity of the weights, which guarantees a compromise between the search for a “better” solution (following the objectives and the given combination of weights) and the feasibility of the individuals ;

- *multiobjective* test cases : to validate PAMUC, a reflexion had first to be done about the norm to use to compare two nondominated sets (obtained by PAMUC and a classical weighted sum method, the latter approach being widely spread in industrial applications). Then, after *R1*-norm was chosen, 9 multiobjective problems were tested, and showed that PAMUC clearly outperforms a linear aggregation of the criteria. This is due to the fact that the fitness function of each individual is not computed independently from the other members of the population, but takes them into account for their ranking. Additionally, considering the satisfaction of the constraints as a new objective and adapting weights at each generation seem to be a suitable way to tackle the constraints. Nevertheless, possible users of PAMUC must be reminded that though performing better than the classical weighted sum method, no claim is made ensuring that it *systematically* finds solutions distributed along the trade-off surface for varying weights.

These various examples illustrate the robustness and efficiency of PAMUC in small size examples (number of variables ≤ 20, number of constraints ≤ 50), constituting thus a well adapted tool for pre-design optimization. The application of PAMUC to industrial mechanical components (namely : space valves) will be presented in Chapter 6.

One of the most interesting advantage of PAMUC is that no tuning of parameters is required. Indeed, after discussion (cf. § 4.4.4), when no additional information is provided by the user, values of $p_i = 1$ and $q_i = 0$ are recommended for all criteria $i = 1,\ldots, m$ (or $m+1$ if there are constraints). Furthermore, the method has demonstrated to be robust with respect to small variations of the weights (cf. § 4.4.5). It is also important to underline that this improvement of the results obtained thanks to PAMUC needs only a reasonable supplementary amount of time in comparison with the weighted sum method (less than 50% in the most unfavourable example treated in this work : cf. § 4.4.6).

Finally, the methodology of conceiving PAMUC still remained general, i.e. no particular hypothesis had to be done about the nature of the variables (hence the codings), the objectives or the constraints, letting the door open to the next step : the incorporation of expert rules within the optimization procedure.

CHAPTER 5 – EXPERT RULES FOR MECHANICAL DESIGN OPTIMIZATION

5.1 Introduction

The second part of this work adresses the pre-design optimization problem where not only geometrical and material variables are involved, but also topological ones. As it implies a dramatic increase of the size of the search space – since designs with very different configurations are shuffled altogether within the evolutionary algorithm –, a standard evolutionary algorithm, without using any additional knowledge, would have hindrances to reach the feasible domain. More fundamentally, when engineers have to choose the best design among a set of proposals, they do not use merely numerical models of the physical behaviour of structures and materials. They also take technological requirements into account, which can seldom be modelled by mathematical equations, but are often predominent in the final choice for one design against another. Those technological constraints are more naturally translated in terms of *rules*.

Considering knowledge as rules has been used since the late sixties in expert systems, which gather a collection of expert rules from scientists in a specific field. Though they have provided outstanding results in some applications, they have severe limitations, and are restricted to narrow domains of applications (§ 5.2). Therefore, a general approach, PAMUC II, mingling multicriteria evolutionary optimization (the PAMUC method introduced in Chapter 3) and expert rules will be presented in § 5.3.

Then, PAMUC II will be validated for single-objective and multiobjective examples with rules, in particular on mechanical benchmarks, and special issues (computational time, consistency of the rule base) will be discussed (§ 5.4), followed by the conclusions (§ 5.5).

5.2 Knowledge-based systems and expert rules for design optimization

5.2.1 Historical background

In order to take into account knowledge from experts, programmers have developed since the late sixties softwares called *expert systems*. Their aim is to reproduce the reasoning that human experts would make in a specific domain, enabling thus to solve complex problems.

The birth of expert systems has to be replaced in a more general stream : artificial intelligence (AI). Artificial intelligence is a constituent part of computer sciences that appeared in the 1950's. Charniak and McDermott consider that modern AI was born in 1956 [CHA85], when the Dartmouth conference (organized by John McCarthy and Marvin Minsky) took place, wherein the expression "artificial intelligence" was used for the first time. Following Savory, AI is the set of computer techniques simulating some of the human natural abilities [SAV88]. AI deals with problems as :

- automatic theorem proving [FIT90] ;
- natural language understanding ;
- speech processing ;
- vision and robotics ;
- expert systems and knowledge acquisition ;
- representation of knowledge.

AI is based on the fundamental hypothesis that what human brain does can be considered, until a certain level, as a computation [TEL88]. In expert systems, knowledge is based on logics, e.g. propositional or predicate calculus. The first and most representative expert systems are DENDRAL, MACSYMA, PROSPECTOR, etc. [HAY83,OLS92]. As MYCIN illustrates many features of expert systems, it will be briefly discussed hereafter.

Though the particular goal of MYCIN is to make easier the diagnosis of infectious diseases, its approach was used in many other expert systems. The context of MYCIN development is the following : when patients have just undergone a surgical operation, they may become victims of infections which have to be eliminated immediately. In such a situation, asking advice to one (or several) physicians(s) is not always possible. Therefore, to provide an expert opinion in any case, MYCIN was created.

The principle of MYCIN is based on the backward-chaining algorithm [SAV88] : it means that it starts from the goal to be obtained (in this case : the antibiotic to be prescribed) ; then, a therapy hypothesis is chosen, and an induction reasoning is performed to find the conditions which have to be verified in order to validate the prescription. From the knowledge acquired thanks to answers given by the user (to judicious questions asked by the software), MYCIN is able to deduce new information (by forward-chaining). This combination of induction and deduction is called mixt-chaining.

Besides, each rule has an uncertainty, which depends on statistics related to the diagnoses. Each solution proposed by the program is therefore given with a certain probability of exactness.

Though MYCIN did furnish excellent results, this approach has never been completely accepted by members of the medical community, and it is interesting to note why they were so reticent to use it. First, the computational cost related to 1970 computers restricted the practical application in a real-life context. Moreover, even if many parameters were taken into account in MYCIN, its behaviour was still over-simple in comparison with the complexity of human symptoms and the huge families of medical treatments. This limitation is crucial, since it underlines that once developed, an expert system is applicable only to a very restricted field, and in specific conditions.

The architecture of classical expert systems is divided in three parts [DEL87,DEL88] :

- the *knowledge base*, contains the values of the variables (§ 5.2.1.1) ;
- the *rule base*, composed of all the rules synthetizing the expert knowledge (§ 5.2.1.2) ;
- the *inference engine*, manipulating the rules (§ 5.2.1.3).

5.2.1.1 Knowledge base

The knowledge base contains the values of the facts (i.e. the variables), which can be :

– boolean : e.g. VARIABLE 1 ≡ PRESENCE OF A HOLE = TRUE ;
– symbolic : e.g. VARIABLE 2 ≡ JOINT MATERIAL = “POLYMER” ;
– integer : e.g. VARIABLE 3 ≡ NUMBER OF BOLTS = 5 ;
– discrete : e.g. VARIABLE 4 ≡ AREA OF CROSS-SECTION OF 2ND BEAM = 10 cm^2 (value taken from a catalogue for instance) ;
– real : e.g. VARIABLE 5 ≡ DIAMETER = 12.5 cm.

5.2.1.2 Rule base

The *rule base* contains the knowledge in terms of expert rules, generally coded as logical expressions written as :

IF condition THEN action.

The shortest way to model expert rules is to use the propositional calculus. In its syntax, rules are fixed, i.e. it is assumed that their content is not modified during the process. Rule grammar was precisely defined under Backus normalized form [DEL87] defined in Tables 5.1 and 5.2. Table 5.1 presents its final elements, used in the fundamental definitions gathered in Table 5.2. Both tables describe thus a complete grammar, based on the propositional calculus, and sufficient to model it. Its keywords are :

{ IF, AND, THEN, NOT },

and two symbols have a specific meaning :

::= indicates that the term preceding this symbol is defined by what follows it ;
/ indicates the alternative.

Final elements	
<BOOLEAN VARIABLE>	the list of boolean variables is provided by the expert
<SYMBOLIC VARIABLE>	the list of symbolic variables is provided by the expert
<INTEGER VARIABLE>	the list of integer variables is provided by the expert
<DISCRETE VARIABLE>	the list of discrete variables is provided by the expert
<REAL VARIABLE>	the list of real variables is provided by the expert
<POSSIBLE SYMBOLIC VALUE>	a symbolic value
<INTEGER NUMBER>	an integer number
<DISCRETE NUMBER>	a discrete number
<REAL NUMBER>	a real number

Table 5.1 : Final elements of the Backus grammar [DEL87].

Definitions	
<RULE> **::=** IF <CONDITION> THEN <CONCLUSION>	
<CONDITION> **::=**	<PREMISS> /
	<PREMISS> AND <CONDITION>

<PREMISS> ::=	<BOOLEAN VARIABLE> /
	NOT <BOOLEAN VARIABLE> /
	<SYMBOLIC VARIABLE> = <SYMBOLIC VALUE> /
	<SYMBOLIC VARIABLE> ≠ <SYMBOLIC VALUE> /
	<INTEGER VARIABLE> <COMPARATOR> < INTEGER VALUE>
	<DISCRETE VARIABLE> <COMPARATOR> <DISCRETE VALUE>
	<REAL VARIABLE> <COMPARATOR> <REAL VALUE>
<SYMBOLIC VALUE> ::=	<SYMBOLIC VARIABLE> /
	<POSSIBLE SYMBOLIC VALUE>
<REAL VALUE> ::=	<REAL VARIABLE> /
	<REAL NUMBER> /
	<REAL VALUE> <OPERATOR> <REAL VALUE>
<DISCRETE VALUE> ::=	<DISCRETE VARIABLE> /
	<DISCRETE NUMBER> /
	<DISCRETE VALUE> <OPERATOR> <DISCRETE VALUE>
<INTEGER VALUE> ::=	<INTEGER VARIABLE> /
	<INTEGER NUMBER> /
	<INTEGER VALUE> <OPERATOR> <INTEGER VALUE>
<COMPARATOR> ::=	< /
	> /
	≤ /
	≥ /
	= /
	≠
<OPERATOR> ::=	+ /
	– /
	* /
	etc.
<CONCLUSION> ::=	<ACTION> /
	<ACTION> AND <CONCLUSION>
<ACTION> ::=	<BOOLEAN VARIABLE> /
	NOT <BOOLEAN VARIABLE> /
	<SYMBOLIC VARIABLE> = <SYMBOLIC VALUE> /
	<INTEGER VARIABLE> = <INTEGER VALUE> /
	<DISCRETE VARIABLE> = <DISCRETE VALUE> /
	<REAL VARIABLE> = <REAL VALUE> /

Table 5.2 : Fundamental definitions in the Backus grammar [DEL87].

Each rule is divided in two parts :

- the *condition*, which corresponds to tests determining whether a rule is susceptible to be applied ;
- the *action*, which is related to assignments of values to variables.

For example, here is a rule with four variables involved :

$$\mathit{IF}\ (\underbrace{\underbrace{\mathit{VAR1} = 0.3}_{\mathit{PREMISS\ 1}}\ \mathit{AND}\ \underbrace{\mathit{VAR2} > 10}_{\mathit{PREMISS\ 2}}}_{\mathit{CONDITION}})\ \mathit{THEN}\ (\underbrace{\underbrace{\mathit{VAR3} = 4}_{\mathit{CONCLUSION\ 1}}\ \mathit{AND}\ \underbrace{\mathit{VAR4} = \mathit{TRUE}}_{\mathit{CONCLUSION\ 2}}}_{\mathit{ACTION}})$$

5.2.1.3 Inference engine

Finally, the third constitutive part of expert systems is the *inference engine*, which manipulates the rules to feed the work memory of the session, by deducing new facts (*deduction*) or by retrieving the conditions that led to a given situation (*induction*).

As already introduced in MYCIN, in propositional systems, three different kinds of inference engines have been developed [DEL88] : backward-chaining (to perform induction), forward-chaining (deduction) and mixt-chaining (combination of forward- and backward-chaining). As in design optimization problem with expert knowledge, one is mostly interested in deducing what rules to apply for a specific design (to check if it respects some technological requirements for instance), a deeper insight will be made on forward-chaining.

The forward-chaining algorithm is based on the MODUS PONENS, a general principle of deduction written as follows :

IF *F1 AND F2 AND … AND FN ARE TRUE,*
AND IF *THE RULE "if F1 and F2 and … and FN, then F" IS TRUE,*

THEN *F IS TRUE*

The corresponding algorithm is converted in pseudo-code in Fig. 5.1.

```
begin
    {initialization}
    declare active all the rules in the rule base
    while (some rules have their conditions satisfied)
        determine the first rule to apply
        execute the rule
        if        (the conclusion of the rule does not contradict
                  any element in the work memory)
        then      execute the conclusion of the rule
                  desactivate the rule
        else
                  base not consistent
        end
    end
end
```

Fig. 5.1 : Forward-chaining algorithm (in pseudo-code) [DEL87].

From the initial data, the forward-chaining algorithm will try to find all the facts that can possibly be deduced. To illustrate the working of this algorithm on a simple example, a graphical representation may be useful : the *AND-OR trees*, where facts are represented by nodes. An example of AND-OR tree is depicted in Fig. 5.2, to symbolize the 6 rules below :

Rule R1 :	$B \Rightarrow A$
Rule R2 :	$C \Rightarrow A$
Rule R3 :	$(D \text{ AND } E) \Rightarrow B$
Rule R4 :	$(F \text{ AND } G) \Rightarrow C$
Rule R5 :	$H \Rightarrow G$
Rule R6 :	$I \Rightarrow G$

For example, if facts F and I are true, the algorithm will progressively deduce that G, C and finally A are true. It will be shown below that the order following which the rules are applied have no consequence on the results (cf. § 5.3.1.2). Incidentally, it is interesting to notice the advantage of the expert system approach, which overtly separate the codings of the knowledge (data and rules) and the inference engine, which allows the user to delete, modify or add rules without changing the whole software, as long as there is no contradiction in the rule base. The consistency of the rule base will be discussed in § 5.4.4.

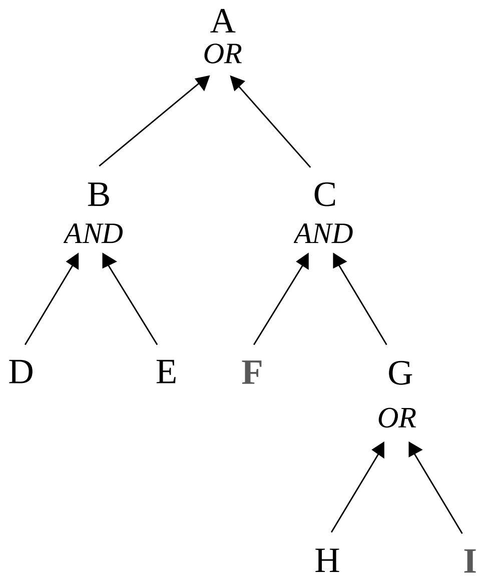


Fig. 5.2 : Illustration of forward-chaining in an AND-OR tree :
from the facts stating that F and I are true, one can deduce that G, C and A are also true.

At the end of the process, the memory has reached a saturated state, i.e. all the rules whose condition parts were satisfied were applied : no other facts could be deduced anymore.

5.2.2 Expert systems for design optimization

The previous section was specially devoted to expound the basics of expert systems, namely what they are able to do and how they work. Here, the emphasis is put on expert systems dedicated to engineering sciences, and particularly structural optimization.

As a matter of fact, most expert systems used for industrial applications have been contrived for scheduling or operations processes, production systems, system planning, etc. [KLE96, MAC02a], because in these fields knowledge can quite easily be modelled by logical rules. Fewer expert systems are directly concerned with structural design optimization. The most emblematic studies in this field are mentioned below.

In [ABE96], Aberšek *et al.* described an expert system (called STATEX) to design and manufacture a gear box. In the first stage of the process, genetic algorithms are used to determine the optimal dimensions of a gear box (with special requirements) ; then, the expert system take technological requirements into account, related to the selection of cutting tool and cutting conditions, the special sequence of machining, the tolerances, etc.

Chau *et al.* also described a knowledge-based system for mechanical design [CHA03], but for liquid-retaining structures. In their expert system, symbolic knowledge based on engineering heuristics in the preliminary stage (e.g. about crack width control) is needed for three types of liquid-retaining structures (a rectangular shape with one compartment, a rectangular shape with two compartments and a circular shape). Thanks to interactive graphical interfaces, the user is directed throughout the design process, which includes preliminary design, load specification, model generation, finite element analysis, code compliance checking and member sizing optimization.

Jiang *et al.* developed another expert system, for the design of scroll compressors used in refrigeration and air conditioners [JIA00]. Indeed, the authors created a visualised solid model of the compressor, which was enhanced by the use of an optimization system. Finite element analysis and expert system strategy were used to study the model, which is useful for improving the quality of manufacturing and assembly accuracy at the later stages. Manufacturability, process planning and cutting tool path code generation were also taken into consideration.

In [KIM99], Kim *et al.* proposed a patchwise optimal layup design method for composite laminates, where the optimal solution is obtained by using an expert system environment combined with a genetic algorithm and a finite element code. In this approach, the weight of composite laminates with ply drop under different loadings is minimized by acting on the stacking sequences and the number of plies in each patch. In this case, the aim of the expert system is to check the number of plies in each patch.

In [NET97], Netten and Vingerhoeds implemented EADOCS (Expert Assisted Design of Composite Sandwich panels) to perform the conception of laminate structures in three phases : the selection of prototype solutions, the selection of concept solutions and their modification. To support these phases, knowledge is necessary. For instance, when designs satisfying the constraints are generated, some rules of thumb can be applied to improve their properties.

Expert systems have also been applied in civil engineering. In [RAM96], in the context of designing industrial roofs, Ramasamy and Rajasekaran compared results obtained by an expert system to solutions given by a genetic algorithm. The expert system contained rules related to the loadings, the temperature, the proportion of the structure in contact with the atmosphere, etc., for various truss structures. Numerical computations showed that in that case, both methods provided very similar results.

The examples cited above are restricted to specific applications. However, in [JON00], Jonson *et al.* addressed the problem of integrating a motion analysis code in a computer-aided design system. It involved the use of a common data model and a special procedure to automate the exchange of data between CAD and motion analysis : this was done thanks to a language based on predicate logic. The authors applied their methodology to a piston-crank mechanism. Logic structure enabled to represent rules acting on the different components of the piston and the crank. The database gathered all the information needed to describe the geometric and functional characteristics of the design.

Finally, in [LEE96], Lee and Kim proposed a unified approach combining a knowledge-based system and a multiobjective hybrid genetic algorithm. The knowledge base plays the role of pre- and post-processor, storing a set of 147 rules separated in 4 categories :

- general rules for input data generation of optimization ;
- rules for input data generation for genetic algorithm and direct search method (ex.: size of the initial population in the GA) ;
- control rules for the GA ;
- rules to select a point among Pareto solutions found by the a posteriori optimization procedure.

This method was applied to the design of a liquefied natural gas carrier ship, and the objectives were to minimize the building and operating costs. The method implemented by Lee and Kim enabled significative enhancements of the design, whilst reducing the computational time.

Thanks to this overview of how expert information was incorporated so far in design problems, the limitations of knowledge-based approaches will be discussed in the next section, compared to classical optimization methods. Then, to solve mechanical pre-design optimization problems, the second version of PAMUC will be presented thoroughly.

5.3 PAMUC II

5.3.1 Preliminaries to the development of PAMUC II

5.3.1.1 Expert systems vs. general optimization methods

The bibliographical study about expert systems devoted to design optimization illustrates their intrinsic limitations. Indeed, their use is generally restricted to narrow applications, making difficult their extension to other problems.

More fundamentally, in optimization context, rules of thumb to guide a search towards a “pseudo-optimal” solution (following some expert rules) may not be efficient, since the algorithm will prefer designs similar to previous ones, instead of exploring the whole search space. Therefore, only expert rules related to constraints that must be satisfied by the design (e.g. for technological reasons) should be kept in the set of rules.

On the other part, general optimization algorithms presented in Chapter 2 – and specially metaheuristics – are well designed to solve larger categories of pure optimization problems, but it is well known that they perform better when additional information is furnished about the problem to solve. Davis summarized this statement in the case of genetic algorithms : “I believe that genetic algorithms are the appropriate algorithms to use in a great many real-world applications. I also believe that one should incorporate real-world knowledge in one’s algorithm by adding it to one’s decoder or by expanding one’s operator set.” (cited in [MIC95a]).

This compromise between the efficiency of a method and its applicability to a wide set of problems is enlightened by the No Free Lunch theorem introduced by Wolpert and Macready [WOL97]. In their study, they proved that if a “black-box” optimization algorithm performs well for a family of problems, it will statistically perform poorer for another one ; in other words, when averaged over *all* possible optimization problems defined over some search space, all black-box algorithms will perform equal [COR03]. This statement has given rise to much controversial debates among the optimization community, mainly based on the classes of functions over which the No Free Lunch theorem holds, and the definition of the set of all functions [IGE03].

However, the following conclusion of this theorem suffers no contradiction : the best way to solve specific applications is to incorporate some knowledge within the optimization process. To take benefit of this statement in design optimization, the idea proposed in this thesis is to enable the user to incorporate knowledge about a particular problem without making the algorithm unapplicable to a larger family of problems. By integrating expert rules as constraints to guide the search, the size of the feasible domain would be reduced, but the core of the algorithm (an EA combined with an expert module able to integrate the user’s rules) would still remain general.

The way rules will be modelled is described in § 5.3.1.2, followed by their integration within EAs in § 5.3.1.3.

5.3.1.2 How to model expert rules ?

To take expert knowledge into account, the methodology is inpired from expert systems. Indeed, the procedure is also separated in three components :

- the knowledge base, containing the values of the variables ;
- the rule base, i.e. the set of expert rules defined by the user ;
- the inference engine (a forward-chaining algorithm).

The rules are written in accordance with the propositional calculus, by means of Backus grammar (cf. § 5.2.1.2). This syntax, albeit simple, is rich enough to express a large variety of information. Furthermore, in comparison with predicate logic, modal logic [RAM88,THA90], or non-monotonic logic [BRE98], propositional (or 0-order) logic provides some interesting theoretical results based on a set of useful theorems [KLE67, RAM88] :

- the *completeness* and *soundness* theorems, stating the Backus grammar is a correct language for propositional calculus ;

- the *decidability* theorem, demonstrating that there is always a mechanical procedure – to apply the rules – that gives a solution ;

- the *confluence* theorem, showing that whatever order of rules is followed, the final result (i.e. the values of all variables at the end of the process) is the same ;

- each application of the rules is performed in a *finite time*, and the asymptotical complexity of a computation is an O(number of rules multiplied by the number of conditions), where the notation $f = O(g)$ means that :

$$\exists c > 0 \text{ and } x_0 \in \mathbb{R} \text{ such that : } \forall x \geq x_0 : f(x) \leq c.g(x). \tag{5.1}$$

These results guarantee that the application of an expert module dealing with the rules can be safely incorporated in the evolutionary algorithm, with no risk of lack of convergence or wrong answer.

5.3.1.3 How to incorporate expert rules within the EA ?

Now that a proper way to model the rules has been proposed, they have to be integrated in the EA. As explained above, the expert rules are considered as requirements which must be fulfilled by the design. In the frame of optimization, it means that rules are supplementary constraints.

Constraint-handling techniques in EAs have been thoroughly discussed in Chapter 3. A family of techniques also dealing with rules is the *repair* strategy, where unfeasible individuals are repaired (with a given probability) following a set of rules. The bibliographical study exposed in § 3.5.1.4 showed that :

- two approaches were encountered in the literature : either the fitness value of the repaired individual is used instead of the fitness of the original one, or the whole individual is replaced (i.e. also its chromosome) ;

- most repair algorithms were designed to solve very restricted applications, by taking specific rules into account, directly depending on the problem, hence related to the coding, the EA parameters, etc. ;

- an additional parameter is introduced : the probability of replacement (p_{rep}), which indicates the rate of members of the population that will be repaired after the generation of new individuals. The contradictory results obtained by different authors clearly demonstrate that no “optimal” value of p_{rep} can be proposed, for it is problem-dependent.

The idea proposed here is to select a subset of N_{subset} members of the population at each generation (with $N_{subset} \approx p_{rep}.N$ since each individual undergoes the expert module with probability p_{rep}, N being the size of the population). Among the N_{subset} individuals, the “bad” ones (i.e. the members of the population which violate the rules defined by the user) are corrected, and replaced in the population. The whole procedure is incorporated in an EA combined with PAMUC, and is described below.

5.3.2 Description of PAMUC II

The key idea in PAMUC II (i.e. PAMUC with an expert module to deal with the rules) is to repair individuals, with a user defined probability. The flow-chart of PAMUC II is shown in Fig. 5.3.

PAMUC II was implemented in the Std-EA written in Matlab (cf. Ch. 4).

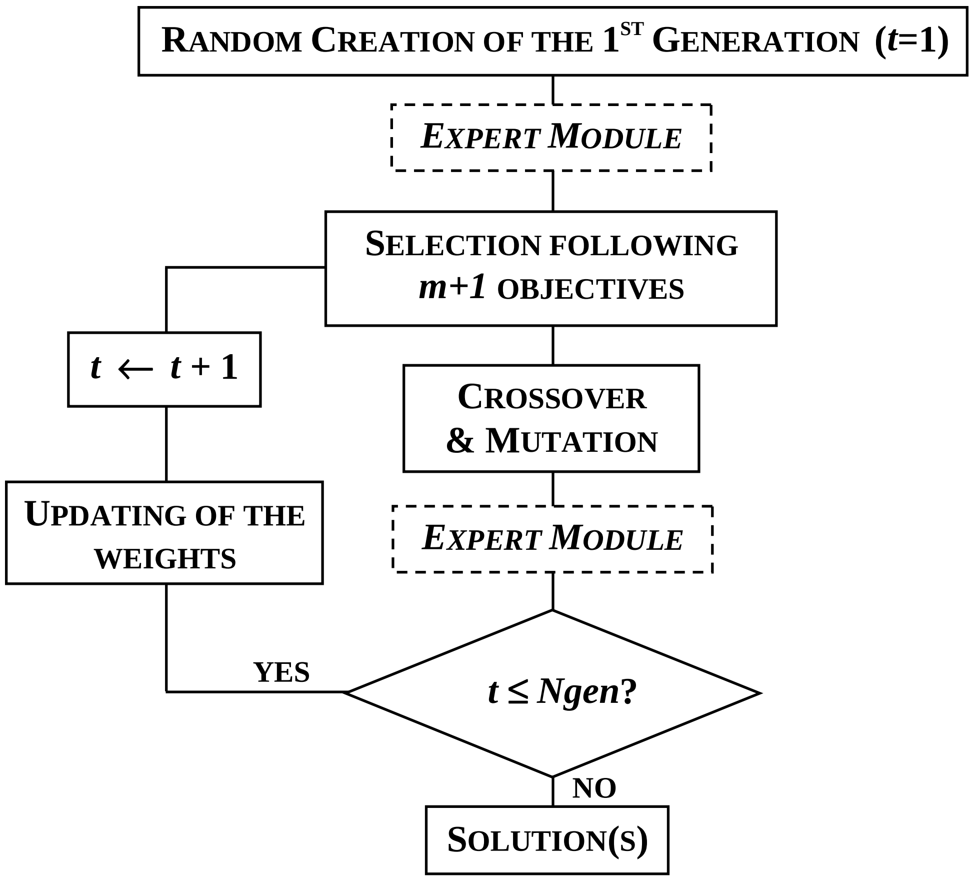


Fig. 5.3 : Flow-chart of PAMUC II.

The architecture is the same as in the PAMUC method, except that after the creation of new individuals (either at the end of the first generation or after the crossover and mutation operators), an expert module is applied for each individual. The algorithm is written in Fig. 5.4 in pseudo-code.

```
begin
    create a random number $0 \le rand \le 1$
    if $rand < p_{rep}$
        declare active all the rules in the rule base
        while (some rules still have their conditions satisfied)
            determine the first rule to apply
            execute the rule
            desactivate the rule
        end
    end
end
```

Fig. 5.4 : Expert module in pseudo-code (p_{rep} is the probability for an individual of being repaired)

The expert module is still composed of three parts (as in expert systems) :

- the *knowledge base*, containing the values of the variables (coded in the chromosome), which may change due to the application of the rules ;
- the *rule base*, containing the rules defined by the user in the beginning of the process ;
- the *inference engine* – a forward-chaining algorithm –, integrated in the core of the algorithm (and thus remaining unchanged whatever rules are added).

A simple example with 2 variables, 2 objectives, 1 constraint and 1 rule will illustrate how PAMUC II works :

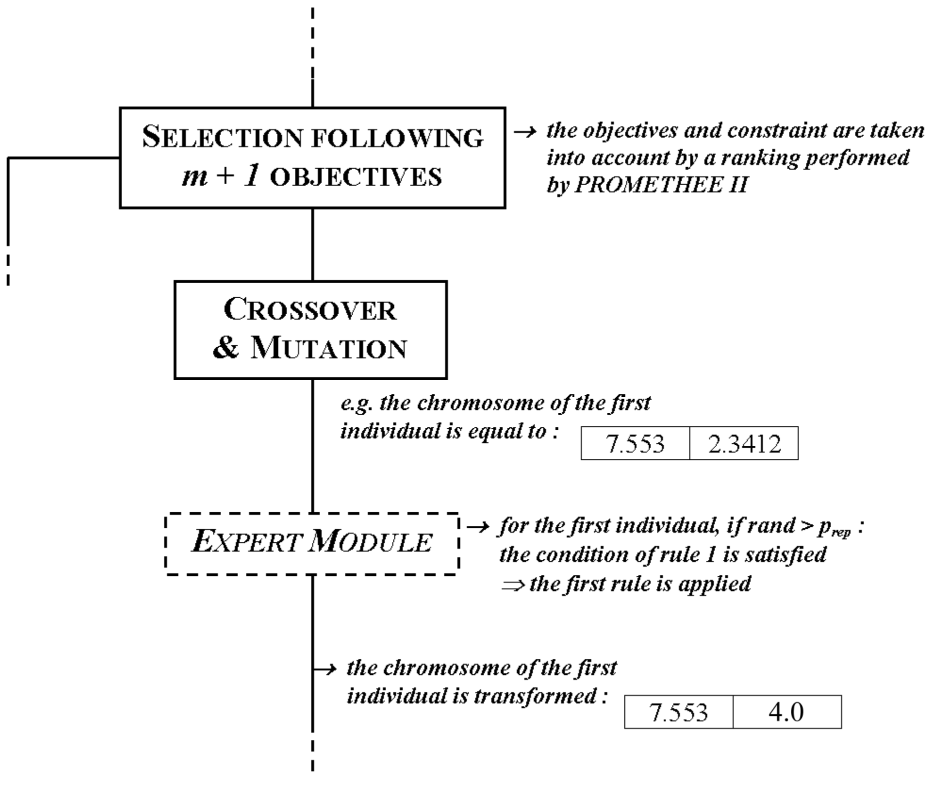
$$max \begin{cases} f_1(\mathbf{x}) = x_1^{\,2} + x_2^{\,2} & (5.2) \\ f_2(\mathbf{x}) = x_1\, x_2 & (5.3) \end{cases}$$

$$\text{subject to : } g_1(\mathbf{x}) = \sin(x_1 + x_2) > 0, \qquad (5.4)$$

$$\text{Rule 1} \equiv \text{if } x_1 \geq 5 \text{ then } x_2 = 4 \qquad (5.5)$$

$$\text{with } 0 \leq x_1 \leq 10 \text{ and } 0 \leq x_2 \leq 10. \qquad (5.6)$$

The multiobjective and constrained aspects are tackled by PAMUC, while Rule 1 is handled by the expert module. As soon as an individual violates the rule, it is corrected and its repaired version replaced it in the population (cf. Fig. 5.5).



SELECTION FOLLOWING m + 1 OBJECTIVES
→ the objectives and constraint are taken into account by a ranking performed by PROMETHEE II
CROSSOVER & MUTATION
e.g. the chromosome of the first individual is equal to :
7.553 | 2.3412
EXPERT MODULE
→ for the first individual, if rand > p_rep : the condition of rule 1 is satisfied ⇒ the first rule is applied
→ the chromosome of the first individual is transformed :
7.553 | 4.0

Fig. 5.5 : Expert module applied to a constrained multiobjective problem with one rule : the first individual of the population is repaired according to Rule 1 (a real-coding is used to build the chromosomes).

5.4 Validation of PAMUC II

5.4.1 Strategy of validation

To validate PAMUC II in solving design optimization problems, it will be compared to the most traditional technique used to deal with the constraints : penalization.

First, to analyze the efficiency of PAMUC II in tackling the rules, it will be compared to single-objective problems with rules. Except for test cases 1 and 2, each single-objective problem will thus be solved by 2 methods :

- the Joines and Houck's penalty method (§ 3.5.1.2), where rules are transformed into mathematical (equality and inequality) constraints ;
- PAMUC II, where the PAMUC procedure is applied for the mathematical constraints and the expert module is used to tackle the rules.

In PAMUC II, as the probability of replacement is a parameter whose value is problem-dependent, computations will be performed with p_{rep} varying from 0 to 100% with a constant step of 5%. If p_{rep} = 35% for instance, 35% of the population (at each generation) will be treated by the expert module whereas PAMUC will be applied to the remaining 65% (with rules transformed into mathematical constraints).

Most of the single-objective problems chosen to validate PAMUC II are taken from mechanical design optimization field. As constraints are very seldom expressed in terms of rules in benchmarks (though it is very common in industrial context), supplementary rules have been added to some examples to increase their complexity. This will be clearly noticed in the presentation of the test cases.

The 7 single-objective problems used to validate PAMUC II are the following :

- two original (mathematical) examples : they are meant to illustrate how PAMUC II works in comparison with PAMUC ;
- one test case due to Hooker *et al.* [HOO00] ;
- four design optimization problems due to Osyczka :
  - a robot gripper [OSY99] ;
  - a beam divided in 6 logs [OSY02] ;
  - a helical spring [OSY02] ;
  - multiple clutch brakes [OSY02].

A comprehensive report of the numerical results is presented in § 5.4.2.

Then, PAMUC II will be applied to three multiobjective problems with rules, in order to check the ability of the method to handle multicriteria optimization as well as constraints and rules. This will be discussed in § 5.4.3.

5.4.2 Single-objective optimization

5.4.2.1 *First test case with rules (TCR 1)*

The first mathematical example built to validate PAMUC II is formulated as follows :

$$\max f(\mathbf{x}) = x_1^2 + x_2^2 - (x_3 - x_4)/x_6 + \exp(x_5/x_7) \tag{5.7}$$

$$\text{subject to : } g_1(\mathbf{x}) = x_1^2 - x_3 - x_5^3 - x_6 + x_7 \geq 0, \tag{5.8}$$

$$\text{rule 1} \equiv \text{if } (x_1^2 - x_2 + x_3) > 5 \text{ then } x_4 = 5, \tag{5.9}$$

$$\text{rule 2} \equiv \text{if } ((x_3 < 4) \vee (x_4 = 5)) \wedge A(R_1) \text{ then } x_5 = 2, \tag{5.10}$$

$$\text{rule 3} \equiv \text{if } (x_5 = 2) \wedge (x_7 \in \{1,3,5\}) \wedge A(R_1,R_2) \text{ then } x_7 = 3, \tag{5.11}$$

$$\text{rule 4} \equiv \text{if } (x_5 < 4) \wedge A(R_1,R_2) \text{ then } x_6 = 3, \tag{5.12}$$

$$\text{with : } x_i \in \{1, 2, 3, 4, 5\} \text{ pour } i = 1, \dots, 7. \tag{5.13}$$

and where $A(R_i)$ means that rule R_i either is not applicable or is desactivated (this guarantees that a rule that can modify a variable x_j is evaluated before rules which use x_j in their condition part).

To apply PAMUC or Joines and Houck's method to the rules, the latter ones must be converted into mathematical constraints. For example, rule 1 is transformed into an inequality constraint (where $g_2(\mathbf{x})$ must be ≥ 0) as expressed in Eq. (5.14). The fact that $g_2(\mathbf{x})$ depends on the value of expression "$(x_1^2 - x_2 + x_3) > 5$" is not a problem for the EA since only the value of $g_2(\mathbf{x})$ is required (and not its sensitivities for instance).

```
rule 1 →   if $(x_1^2 - x_2 + x_3) > 5$ then
               $g_2(\mathbf{x}) = |x_4 - 5|$
           else
               $g_2(\mathbf{x}) = 0$
           end                                          (5.14)
```

The optimum was found by enumeration (in $5^7 = 78125$ evaluations) : $\mathbf{x}^* = (5 ; 5 ; 1 ; 5 ; 2 ; 3 ; 2)$ with $f(\mathbf{x}^*) = 54.0516$. The parameters of the study are gathered in Table 5.3. The algorithm was launched 50 times for each of the 21 values of p_{rep} (from $p_{rep} = 0$ to $p_{rep} = 100\%$ with a constant step of 5%).

Symbol	*Parameter*	*Value*
Env	*Environment*	*Std-EA Matlab*
Cod	*Coding of the variables*	*Decimal*
N	*Size of the population*	*50*
N_{gen}	*Number of generations*	*50*
T_s	*Type of selection*	*Tournament*
n_t	*Number of individuals participating to a tournament*	*2*
p_c	*Probability of crossover*	*1*
T_c	*Type of crossover*	*Uniform*
p_m	*Probability of mutation*	*0.05*
T_m	*Type of mutation*	*Flip*

Table 5.3 : EA parameters for test case TCR 1.

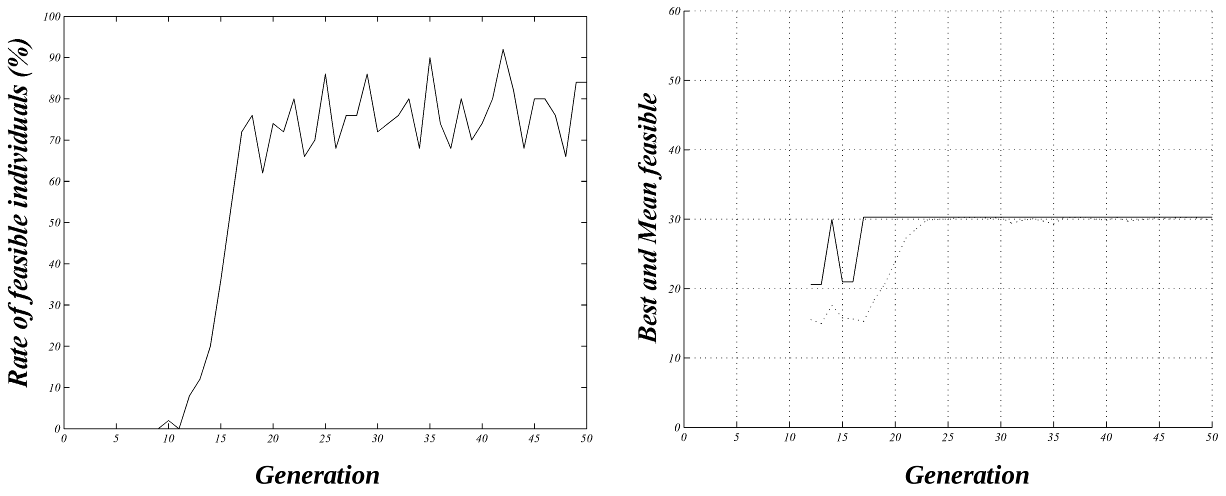


Fig. 5.6 : Rate of feasible individuals (left) and objective function of the best [plain line] and mean [dotted line] of the feasible individuals (right) w.r.t. the generation with $\mathbf{p_{rep} = 0}$ *(for one run) for test case TCR 1.*

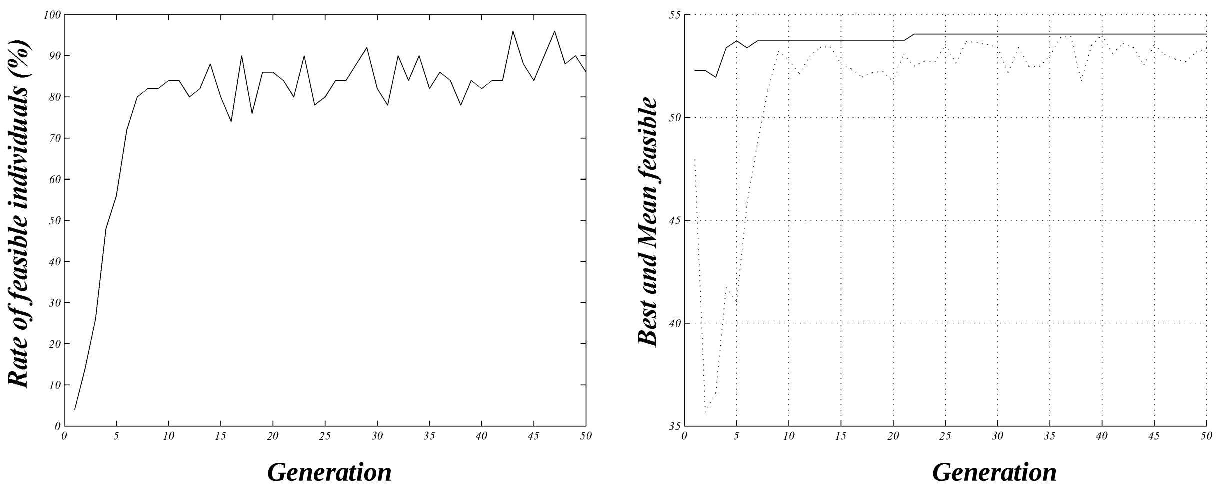


Fig. 5.7 : Rate of feasible individuals (left) and objective function of the best [plain line] and mean [dotted line] of the feasible individuals (right) w.r.t. the generation with $\mathbf{p_{rep} = 0.1}$ *(for one run) for test case TCR 1.*

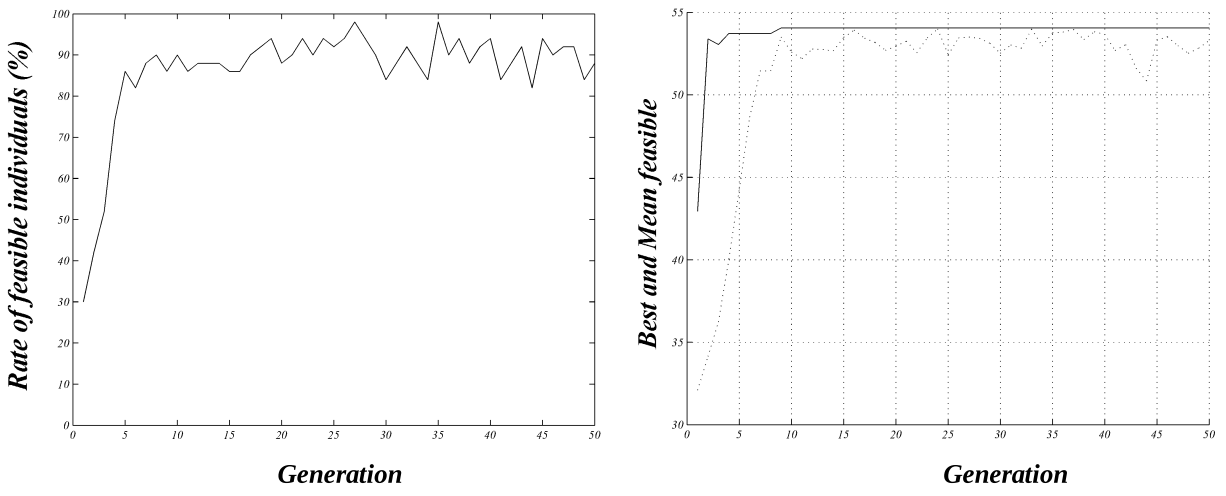


Fig. 5.8 : Rate of feasible individuals (left) and objective function of the best [plain line] and mean [dotted line] of the feasible individuals (right) w.r.t. the generation with $\mathbf{p_{rep} = 0.5}$ *(for one run) for test case TCR 1.*

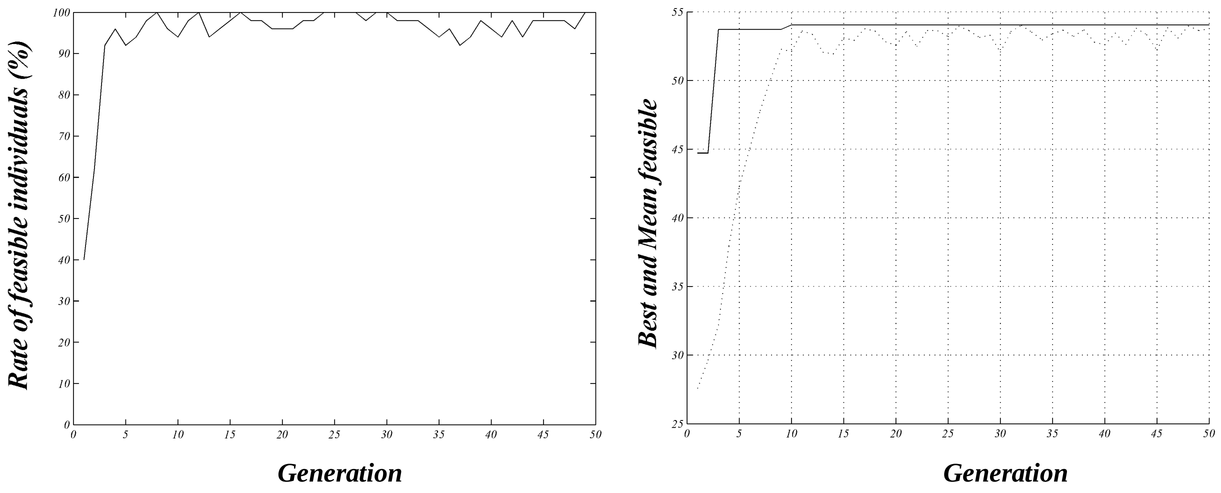


Fig. 5.9 : Rate of feasible individuals (left) and objective function of the best [plain line] and mean [dotted line] of the feasible individuals (right) w.r.t. the generation with $\boldsymbol{p_{rep}} = \boldsymbol{1}$ *(for one run) for test case TCR 1.*

Figures 5.6 to 5.9 exhibit the behaviour of PAMUC II for 4 values of the probability of replacement p_{rep}. The use of the expert module, even with low values of p_{rep}, leads to a faster convergence of the population towards the admissible domain than treating the rules as mathematical constraints, as depicted in Fig. 5.10. Furthermore, the lower value of the best feasible objective function obtained for $p_{rep} = 0$ is due to the fact that without repair, the algorithm sometimes converges to a local maximum (as in Fig. 5.6 [right]).

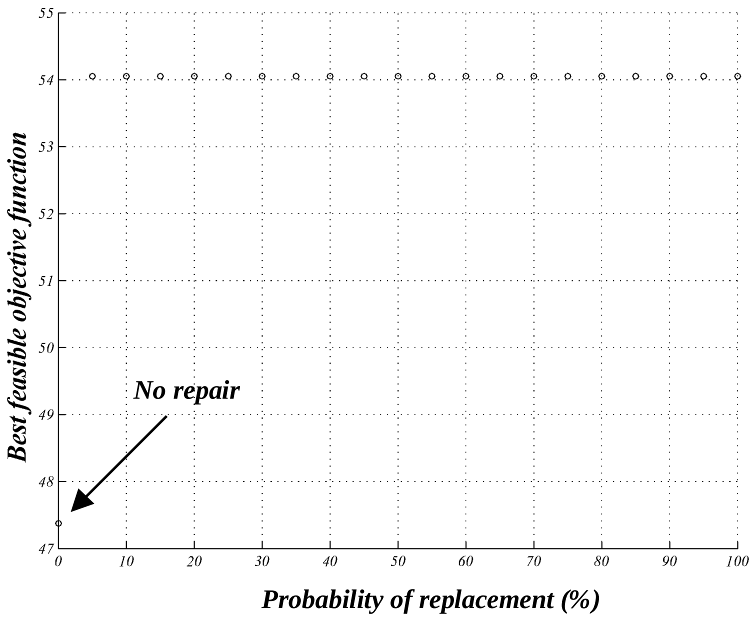


Fig. 5.10 : Mean of the best feasible objective function (over 50 runs) w.r.t. the probability of replacement for test case TCR 1.

Figure 5.11 illustrates the average generation (over 50 runs) needed to reach the global optimum (known exactly because it was calculated by enumeration) with respect to the probability of replacement.

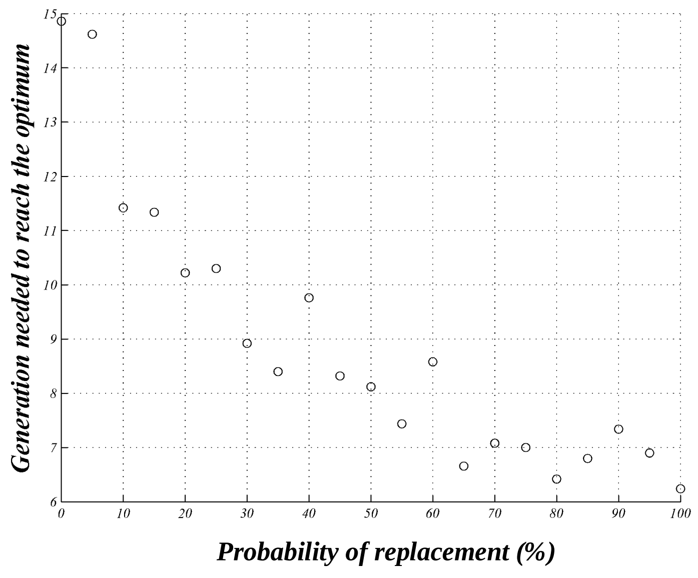


Fig. 5.11 : Average generation (over 50 runs) needed to reach the global optimum w.r.t. the probability of replacement for test case TCR 1.

Increasing p_{rep} enables to diminish drastically the number of generations required to reach the global (admissible) optimum, even if one must keep in mind that the use of the expert module is more expensive than PAMUC alone. A more detailed analysis of the computational time will be performed in § 5.4.5.

5.4.2.2 Second test case with rules (TCR 2)

The second test case with rules is defined as follows :

$$\max f(\mathbf{x}) = x_1^2 + x_2^2 - (x_3 - x_4)/x_6 + \exp(x_5/x_7) \tag{5.15}$$

$$\text{subject to : } g_1(\mathbf{x}) = x_1^2 - x_3 - x_5^3 - x_6 + x_7 \geq 0, \tag{5.16}$$

$$g_2(\mathbf{x}) = 1 - (x_9 + x_{10}\, x_3 - x_8 - 1)^2 \geq 0, \tag{5.17}$$

$$\text{rule 1} \equiv \text{if } (x_1^2 - x_2 + x_3) > 5 \text{ then } x_4 = 5, \tag{5.18}$$

$$\text{rule 2} \equiv \text{if } ((x_3 < 4) \vee (x_4 = 5)) \wedge A(R_1) \text{ then } x_5 = 2, \tag{5.19}$$

$$\text{rule 3} \equiv \text{if } (x_5 = 2) \wedge (x_7 \in \{1,3,5\}) \wedge A(R_1,R_2) \text{ then } x_7 = 3, \tag{5.20}$$

$$\text{rule 4} \equiv \text{if } (x_5 < 4) \wedge A(R_1,R_2) \text{ then } x_6 = 3, \tag{5.21}$$

$$\text{rule 5} \equiv \text{if } (x_7 < x_8) \wedge A(R_1,R_2,R_3) \text{ then } x_9 = x_8, \tag{5.22}$$

$$\text{rule 6} \equiv \text{if } (x_6 \text{ is an odd number}) \wedge A(R_1,R_2,R_4) \text{ then } x_{10} = x_9 + 1, \tag{5.23}$$

$$\text{rule 7} \equiv \text{if } (x_6 \text{ is an even number}) \wedge A(R_1,R_2,R_3) \text{ then } x_{10} = x_9, \tag{5.24}$$

$$\text{with } x_i \in \{1, 2, 3, 4, 5, 6\ 7, 8, 9\} \text{ for } i = 1, \ldots, 10. \tag{5.25}$$

The goal of this test is to illustrate PAMUC II behaviour on a problem similar to TCR 1, but with a larger size of the search space, as well as an increased number of rules and constraints.

The EA parameters are the same as in the previous example. Figure 5.12 shows the objective function of the best feasible individual for different values of p_{rep} (mean over 50 runs), whereas Fig. 5.13 depicts the average generation (over 50 runs) needed to reach the level *f(x)* = 160 with respect to p_{rep}.

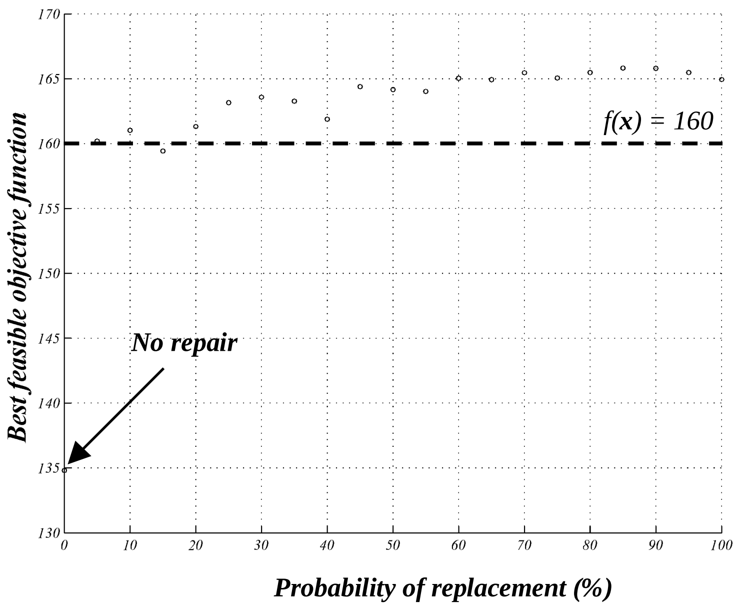


Fig. 5.12 : Mean of the best feasible objective function (over 50 runs) w.r.t. the probability of replacement for test case TCR 2.

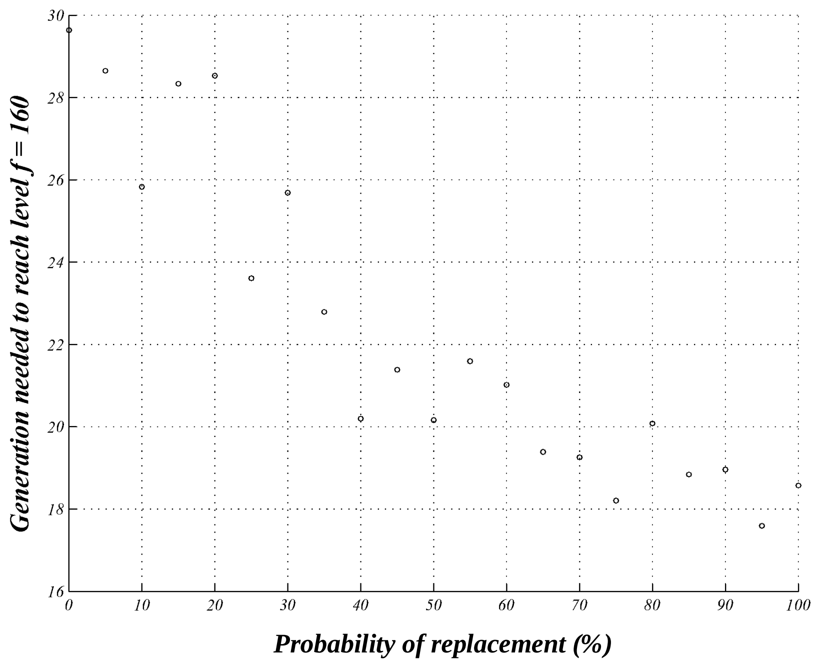


Fig. 5.13 : Average generation (over 50 runs) needed to reach level f(x)=160 w.r.t. the probability of replacement for test case TCR 2.

Once again, it shows that large values of p_{rep} give better results in terms of convergence towards the optimum.

5.4.2.3 Example from Hooker et al. (TCR 3)

The third test case is taken from Hooker *et al.* [HOO00], whose work is mostly devoted to constraint satisfaction problems.

$$\max f(\mathbf{x}) = -(4\,x_1 + 3\,x_2 + 5\,x_3) \tag{5.26}$$

$$\text{subject to : } g_1(\mathbf{x}) = 4\,x_1 + 2\,x_2 + 4\,x_3 - 17 \geq 0, \tag{5.27}$$

$$\text{and } \{\, x_1 , x_2 , x_3 \,\} \text{ all different}, \tag{5.28}$$

$$\text{with } x_i \in \{\, 1, 2, 3, 4, 5 \,\} \text{ for } i = 1, \dots , 3. \tag{5.29}$$

The EA parameters are the same as in the previous example, except that $N = 10$ and $N_{gen} = 10$. The rules compel the variables to be different one from another, hence satisfying automatically the second constraint. Low values of N and N_{gen} were chosen to illustrate the evolution of the results w.r.t. p_{rep} ; indeed, due to the simplicity of the problem (linear constraint and objective function, small search domain), larger values of N and N_{gen} lead to a systematic convergence towards the global optimum $\mathbf{x}^*$ (with $f(\mathbf{x}^*) = -23$).

Fig. 5.14 [right] exhibits a monotonous increase of the averaged best feasible objective function (over 1000 runs).

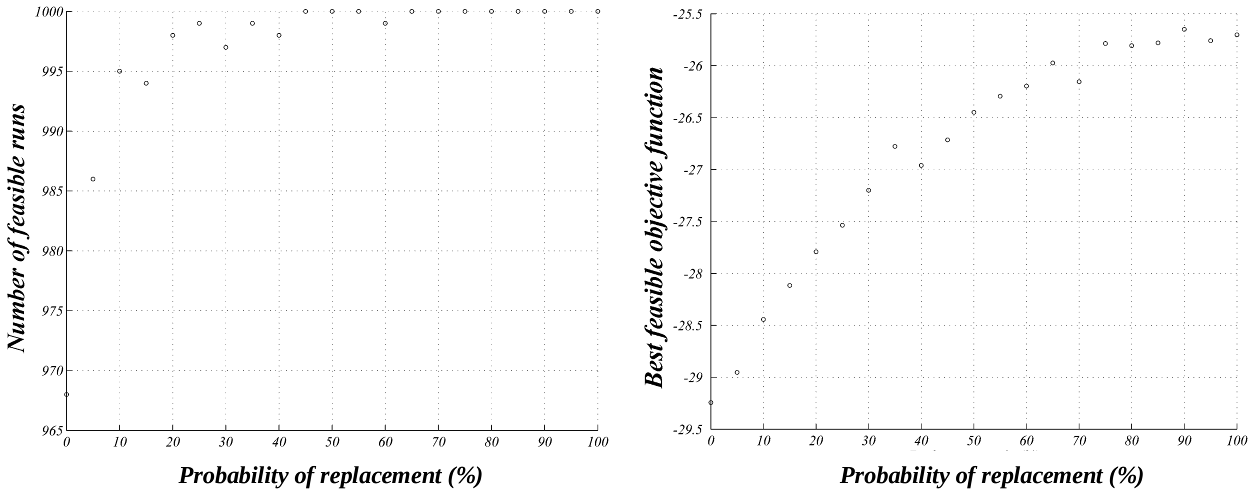


Fig. 5.14 : Evolution of the number of feasible runs (over 1000 runs) [left] and mean of the best feasible objective function (over the feasible runs) [right] w.r.t. the probability of replacement for test case TCR 3.

Results obtained by using only Joines and Houck's penalty method (to deal with both constraints) furnished only about 10% of feasible runs (over 1000 runs), with an average objective value for the best individual (over the feasible runs) of – 32.4, i.e. less than PAMUC alone (cf. Table 5.4).

	Joines and Houck	*PAMUC (no repair)*	*PAMUC II (p_{rep} = 100%)*	*Theoretical Solution*
Number of feasible runs (over 1000 runs)	*112*	*967*	*1000*	*–*
Best feasible objective function at each run (mean over the feasible runs)	*– 32.4*	*– 29.25*	*– 25.7*	*– 23*

Table 5.4 : Comparison of the Joines and Houck's method and PAMUC (II) for problem TCR 3.

5.4.2.4 Robot gripper design problem (TCR 4)

The first benchmark taken from mechanical engineering was devised by [OSY99] (cf. Fig. 5.15).

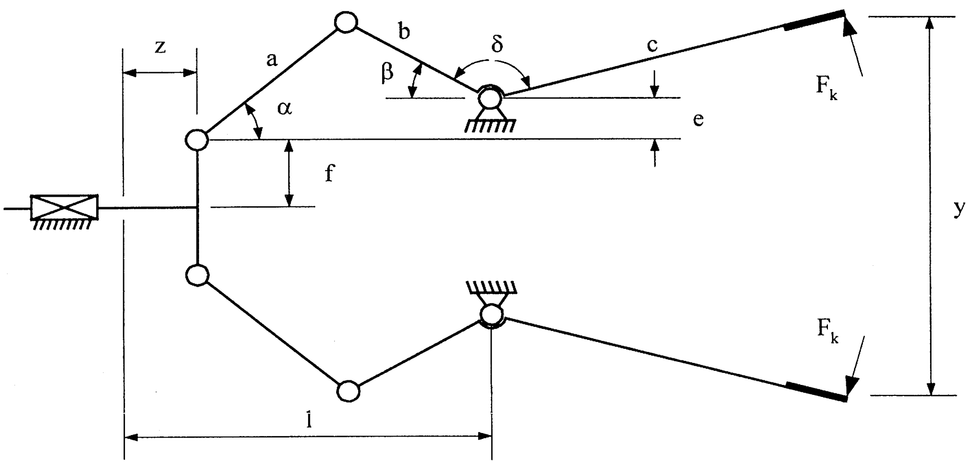


Fig. 5.15 : Robot gripper design problem [OSY99].

The initial benchmark is characterized by 7 continuous variables (a, b, c, e, f, l, δ) and 6 geometrical constraints (relations between angles and lengths, etc.). To make the problem harder to solve, 2 additional constraints (expressed as rules) were introduced by the author (cf. Eqs. (5.37) and (5.38)).

$$\min f_{obj}(\mathbf{x}) = \max_z F_k(\mathbf{x},z) - \min_z F_k(\mathbf{x},z) \tag{5.30}$$

$$\text{subject to : } g_1(\mathbf{x}) = Y_{max} - y(\mathbf{x},Z_{max}) \geq 0, \tag{5.31}$$

$$g_2(\mathbf{x}) = y(\mathbf{x},Z_{max}) \geq 0, \tag{5.32}$$

$$g_3(\mathbf{x}) = y(\mathbf{x},0) - Y_{min} \geq 0, \tag{5.33}$$

$$g_4(\mathbf{x}) = Y_G - y(\mathbf{x},0) \geq 0, \tag{5.34}$$

$$g_5(\mathbf{x}) = (a + b)^2 - l^2 - e^2 \geq 0, \tag{5.35}$$

$$g_6(\mathbf{x}) = (l - Z_{max})^2 + (a - e)^2 - b^2 \geq 0, \tag{5.36}$$

$$\text{rule 1} \equiv \text{if } (a < 4b \text{ and } c < a+b) \text{ then } f = 2e + 10, \tag{5.37}$$

$$\text{rule 2} \equiv \text{if } (a < 4b \text{ and } c \geq a+b) \text{ then } f = e + 50, \tag{5.38}$$

$$\text{with : } 10 \leq a \leq 250,\ 10 \leq b \leq 250,\ 100 \leq c \leq 300, \tag{5.39}$$

$$0 \leq e \leq 50,\ 10 \leq f \leq 250,\ 100 \leq l \leq 300, \tag{5.40}$$

$$1.0 \leq \delta \leq 3.14, \tag{5.41}$$

and where :

$$y(\mathbf{x},z) = 2\,[e + f + c.\sin(\beta + \delta)], \tag{5.42}$$

$$F_k = \frac{Pb.\sin(\alpha + \beta)}{2c.\cos\alpha}, \tag{5.43}$$

$$\alpha = \arccos\left(\frac{a^2 + g^2 - b^2}{2ag}\right) + \phi, \tag{5.44}$$

$$\beta = arc\ cos\left(\frac{b^2 + g^2 - a^2}{2bg}\right) - \phi, \tag{5.45}$$

$$\phi = atan\left(\frac{e}{l - z}\right), \tag{5.46}$$

$$g = \sqrt{(l - z)^2 + e^2}, \tag{5.47}$$

$$P = 100\ [N]; \tag{5.48}$$

$$Y_{min} = 50\ [mm]\ ;\ Y_{max} = 100\ [mm]\ ; \tag{5.49}$$

$$Y_G = 150\ [mm]\ ;\ Z_{max} = 50\ [mm]. \tag{5.50}$$

The objective is to minimize the difference between the maximum and minimum gripping forces needed for the assumed range of gripper end displacement. To compute the minimum and maximum of the forces (which is a 1-variable continuous optimization problem), Nelder and Mead's algorithm (based on the simplex method) from the Matlab toolbox is used [NEL65].

Expert rules are designed in such a way that repaired individuals automatically have constraint 5 and rules 1 and 2 satisfied. The EA parameters are the same as in test case TCR 2, except that a Gray coding is used with 10 bit per variable, and N_{gen} = 100. With Joines and Houck's method, no feasible point was found, whilst results of PAMUC II are greatly enhanced when the probability of replacement is high (cf. Fig. 5.16 and Table 5.5).

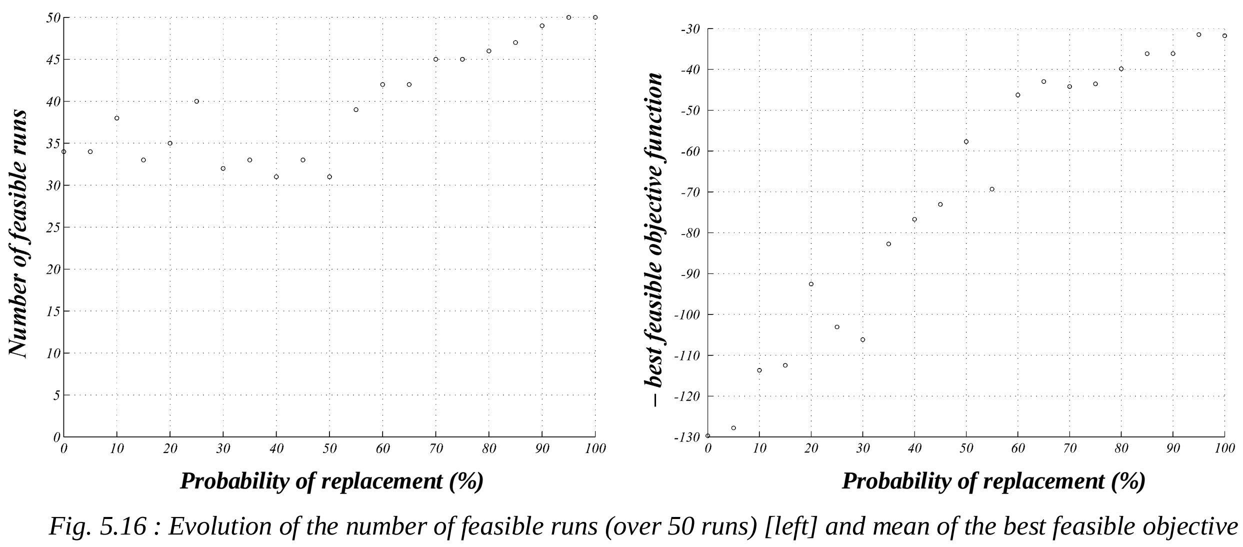


Fig. 5.16 : Evolution of the number of feasible runs (over 50 runs) [left] and mean of the best feasible objective function (over the feasible runs) [right] w.r.t. the probability of replacement for test case TCR 4.

	Joines and Houck	***PAMUC (no repair)***	***PAMUC II (p_{rep} = 100%)***
Number of feasible runs (over 50 runs)	*0*	*34*	*50*
Best feasible objective function at each run (mean over the feasible runs)	*(no feasible solution)*	*129.8*	*31.8*

Table 5.5 : Comparison of the Joines and Houck's method and PAMUC (II) for problem TCR 4 (robot gripper design problem).

5.4.2.5 Beam design problem (TCR 5)

This problem is based on test case S-BDP, already encountered in Chapter 4 (cf. § 4.4.2.7). Here, only the first objective function is optimized (i.e. minimizing the volume of the beam), and 2 supplementary constraints have been added by the author to increase the problem difficulty :

$$\textit{rule 1} \equiv \textit{if } (x_6 > 28 \textit{ mm}) \textit{ then } x_5 = x_6\,, \tag{5.51}$$
$$\textit{rule 2} \equiv \textit{if } (x_5 = x_6) \textit{ then } x_4 = x_5 - 2 \textit{ mm}. \tag{5.52}$$

The other constraints impose that the normal stress must not overstep a critical level, and that each piece of beam should be inferior in height to the following one (cf. Fig. 5.17). Expert rules used with PAMUC II enable to fulfil those latter constraints, as well as rules 1 and 2.

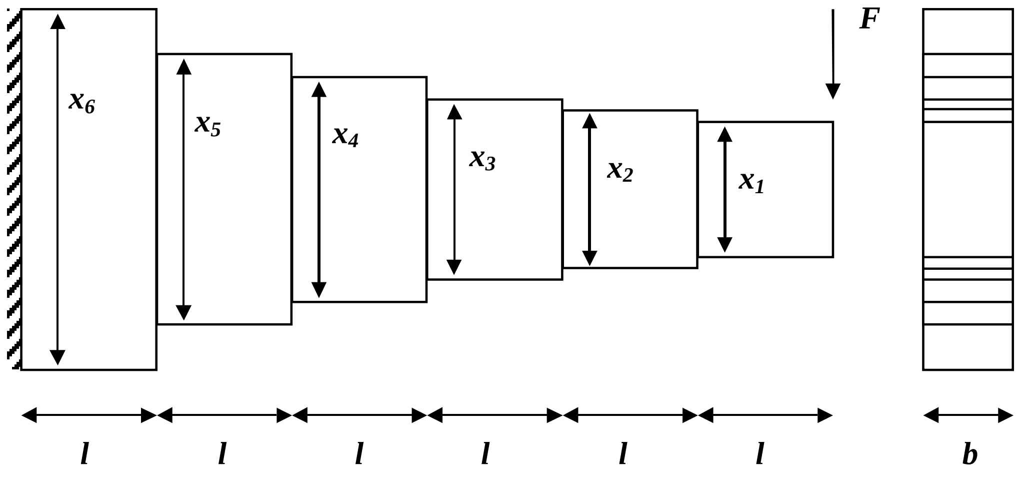


Fig. 5.17 : Beam design problem (adapted from [OSY02]).

The EA parameters are the same as in the previous example, except that N = 100 and N_{gen} = 150. Furthermore, to build the chromosomes, a binary coding was used.

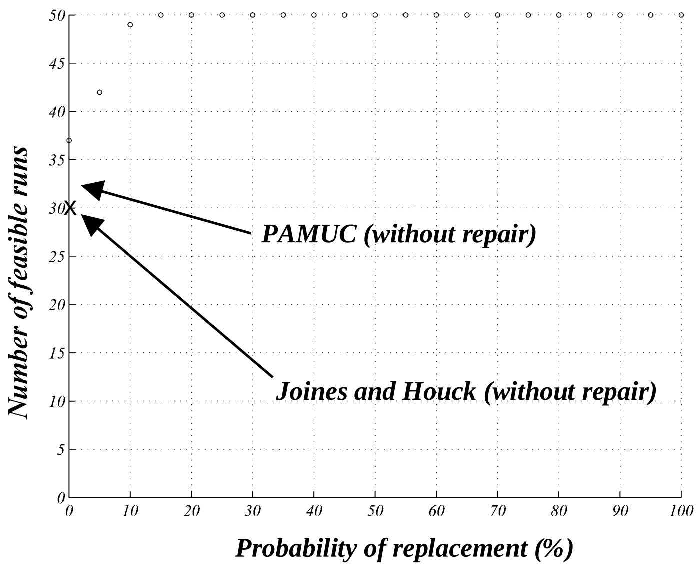


Fig. 5.18 : Evolution of the number of feasible runs (over 50 runs) w.r.t. the probability of replacement for test case TCR 5.

Figure 5.18 illustrates that for values of p_{rep} above 20%, all runs are feasible. Numerical results are gathered in Table 5.6.

	Joines and Houck	*PAMUC (no repair)*	*PAMUC II (p_{rep} = 100%)*
Number of feasible runs (over 50 runs)	*35*	*37*	*50*
Best feasible objective function at each run (mean over the feasible runs)	*$4.22.10^5$*	*$3.8.10^5$*	*$3.8.10^5$*

Table 5.6 : Comparison of the Joines and Houck's method and PAMUC (II) for problem TCR 5 (beam design problem).

To realize how efficient PAMUC II is – compared to a dynamic penalty method without repair –, a plot of the violation rate of the 8th constraint is drawn in Fig. 5.19 for both methods (with p_{rep} = 0.8 in PAMUC II) for one run (constraint 8 imposes x_2 to b greater than x_1). One can see that the number of individuals violating constraint 8 is much smaller when the expert module is applied.

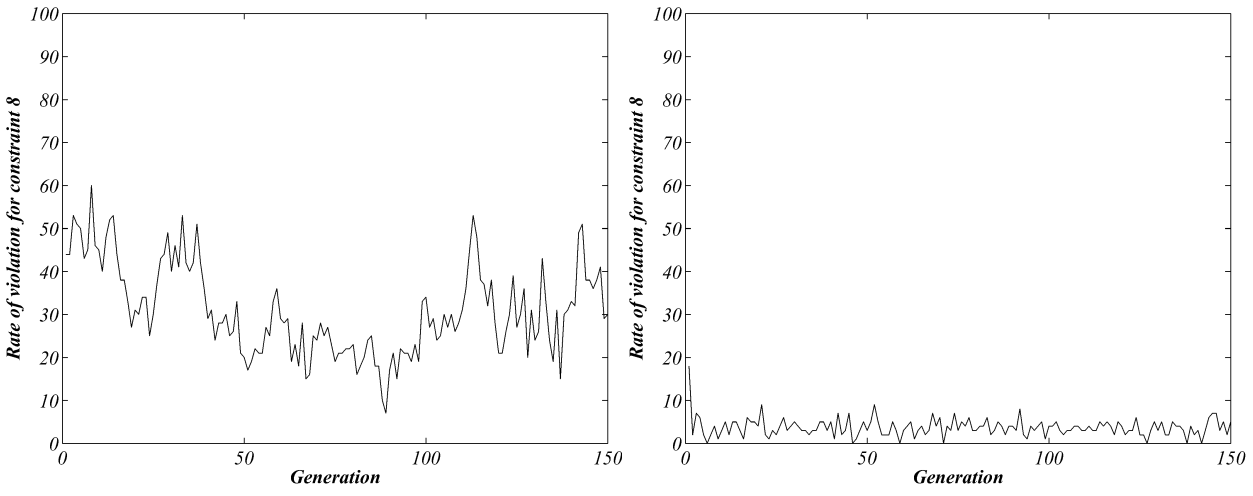


Fig. 5.19 : Evolution of the rate of individuals violating constraint 8 w.r.t. the generation for test case TCR 5 (for one run), using Joines and Houck's method [left] and PAMUC II with p_{rep} = 0.8 [right].

5.4.2.6 Helical spring design problem (TCR 6)

Test case TCR 6 deals with the optimization of a helical spring [OSY02] (cf. Fig. 5.20). Four discrete variables are involved (x_1, x_2, x_3 and the number of coils x_4), and the goal is to minimize the volume. Formulae giving the constraints are based upon Polish Standard PN-85/M-80701-3, describing the procedure to design springs.

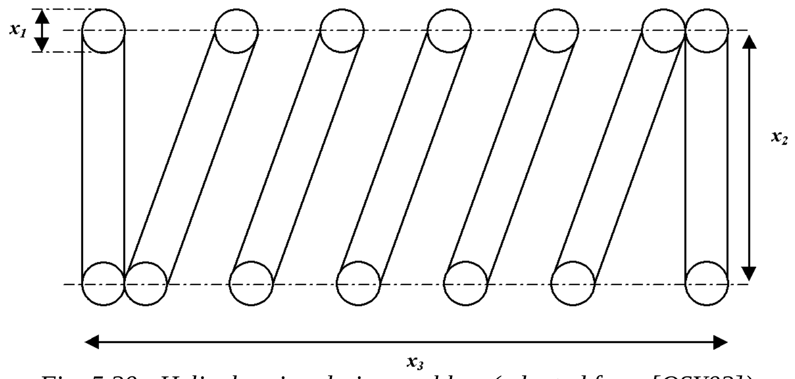


Fig. 5.20 : Helical spring design problem (adapted from [OSY02]).

The optimization problem is written as follows :

$$min\ f(\mathbf{x}) = \frac{\pi^2}{2} x_1^2 \sqrt{x_2^2 x_4^2 + x_3^2} + \frac{\pi}{4} x_1^2 x_2 \tag{5.53}$$

$$subject\ to :\ g_1(\mathbf{x}) = s_f\, \tau_{dop} - k\, \frac{8x_2 P}{\pi\, x_1^3} \geq 0, \tag{5.54}$$

$$g_2(\mathbf{x}) = \Delta c.c - \left| c - \frac{Gx_1^4}{8x_2^3 x_4} \right| \geq 0, \tag{5.55}$$

$$g_3(\mathbf{x}) = x_3 - \frac{8x_2^3 x_4 P}{Gx_1^4} - x_1\, x_4\, (1 + \alpha) \geq 0, \tag{5.56}$$

$$g_4(\mathbf{x}) = \eta_{dop} - \frac{800 x_2^3 x_4 P}{Gx_3 x_1^4} \geq 0, \tag{5.57}$$

$$g_5(\mathbf{x}) = 7\, x_1 - x_2\, , \tag{5.58}$$

$$g_6(\mathbf{x}) = 60 - x_2\, , \tag{5.59}$$

$$where :\quad \tau_{dop} = 605.0\ [N/mm^2] = allowable\ shear\ stress, \tag{5.60}$$

$$P = 1850\ [N] = load, \tag{5.61}$$

$$k = 1 + \frac{5}{4}\left(\frac{1}{w}\right) + \frac{7}{8}\left(\frac{1}{w}\right)^2 + \left(\frac{1}{w}\right)^3 = Wahl\ factor, \tag{5.62}$$

$$s_f = 1.12 = coeff.\ of\ allowable\ changes\ in\ shear\ stress, \tag{5.63}$$

$$w = x_2/x_1 = index\ of\ the\ spring, \tag{5.64}$$

$$c = 20.55\ [N/mm] = required\ stiffness\ of\ the\ spring, \tag{5.65}$$

$$\Delta c = allowable\ deviation\ of\ the\ stiffness = 3\%\ , \tag{5.66}$$

$$G = 81400\ [N/mm^2] = modulus\ of\ rigidity, \tag{5.67}$$

$$\begin{aligned} \alpha = \ & -3.10^{-5}\, w^3 + 0.25.10^{-2}\, w^2 - 0.027\, w \\ & + 0.139 \quad if\ x_1 < 0.8, \end{aligned} \tag{5.68}$$

$$\begin{aligned} \alpha = \ & -2.10^{-5}\, w^3 + 0.2.10^{-2}\, w^2 - 0.18.10^{-2}\, w \\ & + 0.0627 \quad if\ x_1 \geq 0.8, \end{aligned} \tag{5.69}$$

$$\lambda = x_3/x_2 = spring\ slenderness\ ratio, \tag{5.70}$$

$$\begin{aligned} \eta_{dop}\ & = spring\ allowable\ flexibility\ ratio \\ & = 2.10^{-5}\, \lambda^6 + 0.0033\, \lambda^5 - 0.0399\, \lambda^4 + 0.087\, \lambda^3 \\ & \quad +0.8587\ \lambda^2 + 0.9852\, \lambda + 71.203, \end{aligned} \tag{5.71}$$

$$x_1 \in \{ 4 ; 4.5 ; 5 ; 5.5 ;\ldots ; 12 \}, \quad (5.72)$$
$$x_2 \in \{ 30 ; 31 ; 32 ; 33 ;\ldots ; 90 \}, \quad (5.73)$$
$$x_3 \in \{ 100 ; 102 ; 104 ; 106 ;\ldots ; 300 \}, \quad (5.74)$$
$$x_4 \in \{ 5 ; 5.5 ; 6 ; 6.5 ;\ldots ; 14 \}. \quad (5.75)$$

Two additional constraints have been introduced by the author to make the problem harder to solve :

$$g_7(\mathbf{x}) = 4\, x_2 - x_3 \geq 0, \quad (5.76)$$
$$\text{rule 1} \equiv \text{if } (\lambda < 2 \text{ or } \lambda > 5) \text{ then } x_4 \notin [6,10]. \quad (5.77)$$

Individuals undergoing a trip in the expert module automatically respect constraints 5 to 7 and rule 1. The EA parameters are the same as in the previous example except that $N = 50$ and $N_{gen} = 100$. Statistics on 50 runs are gathered in Table 5.7, and show that Joines and Houck's technique furnished 10% of feasible runs, while PAMUC II gave 100% of feasible runs. Figure 5.21 shows that even without repair, PAMUC is still better than the dynamic penalty method. It is also clear that the quality of the solution increases with p_{rep}.

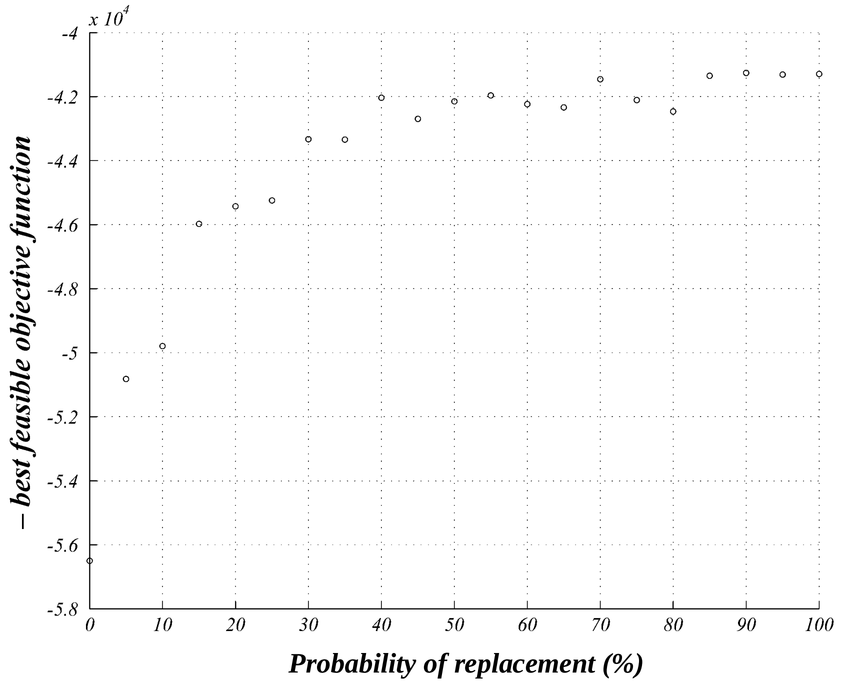


Fig. 5.21 : Evolution of the best feasible objective function (over 50 runs) w.r.t. the probability of replacement for test case TCR 6.

	Joines and Houck	**PAMUC (no repair)**	**PAMUC II (p_{rep} = 100%)**
Number of feasible runs (over 50 runs)	*5*	*50*	*50*
Best feasible objective function at each run (mean over the feasible runs)	*$10.3.10^4$*	*$5.65.10^4$*	*$4.13.10^4$*

Table 5.7 : Comparison of the Joines and Houck's method and PAMUC (II) for problem TCR 6 (helical spring design problem).

5.4.2.7 *Multiple clutch brakes design problem (TCR 7)*

The last single-objective example with rules presented in this study is a multiple clutch brakes design problem [OSY02] (cf. Fig. 5.22).

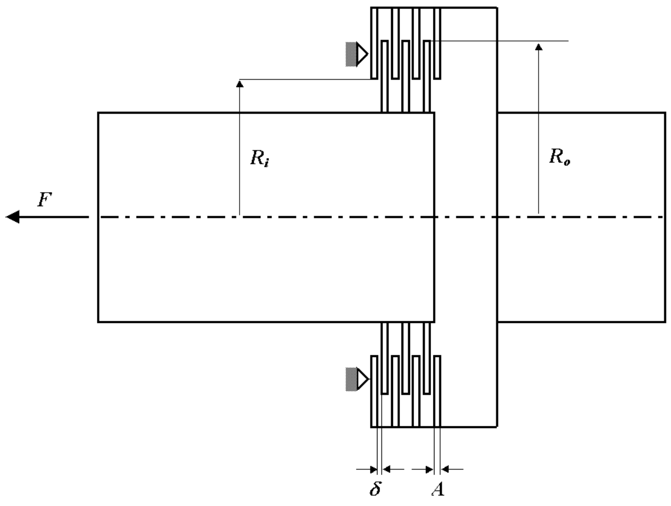


Fig. 5.22 : Multiple clutch brakes design problem (adapted from [OSY02]).

Five discrete variables describe the design : R_i, R_0, A, δ and Z (i.e. the number of friction surfaces). The problem is formulated as follows :

$$\min f(\mathbf{x}) = \pi (R_0^{\,2} - R_i^{\,2})\, A\, (Z+1)\, \rho \tag{5.78}$$

subject to :

$$g_1(\mathbf{x}) = R_0 - R_i - \Delta R \geq 0, \tag{5.79}$$
$$g_2(\mathbf{x}) = L_{max} - (Z+1)(A+\delta) \geq 0, \tag{5.80}$$
$$g_3(\mathbf{x}) = p_{max} - p_{rz} \geq 0, \tag{5.81}$$
$$g_4(\mathbf{x}) = p_{max}\, v_{sr\,max} - p_{rz}\, v_{sr} \geq 0, \tag{5.82}$$
$$g_5(\mathbf{x}) = v_{sr\,max} - v_{sr} \geq 0, \tag{5.83}$$
$$g_6(\mathbf{x}) = t_{max} - t_h \geq 0, \tag{5.84}$$
$$g_7(\mathbf{x}) = M_h - s\, M_s\, v_{sr} \geq 0, \tag{5.85}$$
$$g_8(\mathbf{x}) = t_h \geq 0, \tag{5.86}$$

where :

$$\Delta R = 20\ [mm] = \text{minimum difference between radii}, \tag{5.87}$$
$$A_{max} = 3.0\ [mm] = \text{maximum disc thickness}, \tag{5.88}$$
$$A_{min} = 1.5\ [mm] = \text{minimum disc thickness}, \tag{5.89}$$
$$L_{max} = 30\ [mm] = \text{maximum length}, \tag{5.90}$$
$$Z_{max} = 10 = \text{maximum number of discs}, \tag{5.91}$$
$$v_{sr\,max} = 10\ [m/s] = \text{max. relative speed of the lipstick}, \tag{5.92}$$
$$\mu = 0.5 = \text{coefficient of friction}, \tag{5.93}$$
$$\rho = 7.8.10^{-6}\ [kg/mm^3] = \text{density of material}, \tag{5.94}$$
$$s = 1.5 = \text{factor of safety}, \tag{5.95}$$
$$M_s = 40\ [Nm] = \text{static input torque}, \tag{5.96}$$
$$M_f = 3\ [Nm] = \text{frictional resistance torque}, \tag{5.97}$$
$$n = 250\ [rpm] = \text{input speed}, \tag{5.98}$$
$$p_{max} = 1\ [MPa] = \text{max. allowable pressure on the disc}, \tag{5.99}$$
$$J_z = 55\ [kg.mm^2] = \text{moment of inertia}, \tag{5.100}$$
$$t_{max} = 15\ [s] = \text{maximum stopping time}, \tag{5.101}$$

$$F_{max} = 1000\ [N] = \text{maximum actuating force}, \qquad (5.102)$$
$$R_{i\,min} = 55\ [mm], \qquad (5.103)$$
$$R_{0\,max} = 110\ [mm], \qquad (5.104)$$

and :
$$M_h = \text{braking torque} = \frac{2}{3}\mu F Z \frac{R_0^3 - R_i^3}{R_0^2 - R_i^2}, \qquad (5.105)$$

$$p_{rz} = F/S = \frac{F}{\pi(R_0^2 - R_i^2)}, \qquad (5.106)$$

$$v_{sr} = \frac{2\pi n\left(R_0^3 - R_i^3\right)}{90(R_0^2 - R_i^2)}, \qquad (5.107)$$

$$t_h = \frac{J_z \pi n}{30(M_h + M_f)} \qquad (5.108)$$

with :
$$R_i \in \{ 60 ; 61 ; 62 ; \dots ; 80 \}, \qquad (5.109)$$
$$R_0 \in \{ 90 ; 91 ; 92 ; \dots ; 110 \}, \qquad (5.110)$$
$$A \in \{ 1 ; 1.5 ; 2 ; 2.5 ; 3 \}, \qquad (5.112)$$
$$\delta \in \{ 600 ; 610 ; 620 ; \dots ; 1000 \}, \qquad (5.113)$$
$$Z \in \{ 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 \}, \qquad (5.114)$$

Two supplementary rules were added to increase the difficulty of the problem :

$$\text{rule 1} \equiv \text{if } F \leq 800 \text{ N then } R_0 > R_i + 30 \text{ mm}, \qquad (5.115)$$

$$\text{rule 2} \equiv \text{if } A/Ro < 2 \text{ or } A/Ro > 5 \text{ then } Z \in [6,10] \qquad (5.116)$$

The expert module compel the individuals undergoing repair to respect constraint 1 and rules 1 and 2. To satisfy rule 1 for instance, when $F \leq 800$ N and R_0 value is less than R_i + 30 mm (cf. Eq. (5.116)), it is replaced by $R_0^{\,repaired}$, which is created randomly such that $R_0^{\,repaired} \in \{ 90 ; 91 ; \dots ; 110 \}$ and :

$$R_i + 30 \text{ mm} \leq R_0^{\,repaired} \leq R_{0\,max} = 110 \text{ mm} \qquad (5.117)$$

The EA parameters are the same as in the previous example except that N_{gen} = 50. Statistics on 50 runs are gathered in Table 5.8, and show that Joines and Houck's technique furnished 10% of feasible runs (among which the average best objective function is equal to 0.9649), while PAMUC II gave 100% of feasible runs. Furthermore, as illustrated in Figure 5.23, the results are better with values of p_{rep} above 50%.

	Joines and Houck	***PAMUC (no repair)***	***PAMUC II (p_{rep} = 100%)***
Number of feasible runs (over 50 runs)	*5*	*50*	*50*
Best feasible objective function at each run (mean over the feasible runs)	*0.9649*	*0.516*	*0.357*

Table 5.8 : Comparison of the Joines and Houck's method and PAMUC (II) for problem TCR 7 (multiple clutch brakes design problem).

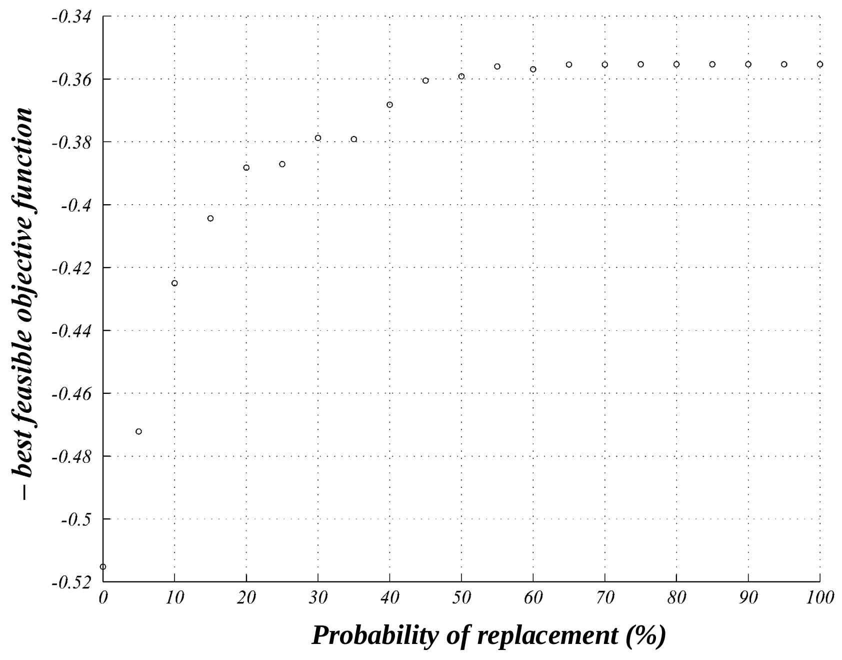


Fig. 5.23 : Evolution of the best feasible objective function (over 50 runs) w.r.t. the probability of replacement for test case TCR 7.

5.4.2.8 Value of the probability of replacement

Although a thorough analysis of the results obtained thanks to PAMUC II for single-objective problems with rules will take place in § 5.4.6, some remarks can already be made about the new parameter introduced in PAMUC II, namely the probability of repair p_{rep}. Indeed, numerical results presented above showed that in *all* test cases, a value of 100% gave the best results, in terms of feasibility and quality of the solution (i.e. the value of the objective function for the best feasible individual).

However, in repair algorithms, no optimal value of p_{rep} suits in all situations (cf. § 3.5.1.4). Indeed, in some applications, a too large value of p_{rep} is likely to prevent the algorithm from exploring widely the search space, guiding it too early towards a narrow region. Therefore, even if PAMUC II seems to perform better with p_{rep} = 100%, a tuning of this parameter should *always* be done when new examples are investigated.

5.4.3 Multiobjective optimization

So far only single-objective optimization test cases with rules have been studied. The final aim of this validation is to demonstrate that PAMUC II is relevant for multiobjective problems with expert rules.

Three multiobjectives examples will be analyzed by applying two methods :

- a classical weighted sum method, combined with Joines and Houck’s penalty technique to deal with both mathematical constraints and expert rules. This choice was guided by the fact that in industrial context, to solve this kind of problem, this would be the most general and widely used approach ;

- PAMUC II, i.e. PAMUC to handle the multiple objectives and mathematical constraints, and the expert module to tackle the rules.

As in Chapter 4, both methods will be applied for different values of the weights, and compared thanks to *R1*-norm (cf. § 4.3). The three applications are 2-objective variants of mechanical component design problems mentioned above :

- the robot gripper design (cf. § 5.4.2.4), with the second objective function being the force transmission ratio (to be minimized) :

$$f_2(\mathbf{x})= \frac{P}{\min\limits_{z} F_k(\mathbf{x},z)}; \tag{5.118}$$

- the beam design (§ 5.4.2.5), where the second objective function is the displacement at the right extremity of the beam :

$$f_2(\mathbf{x})= \frac{Fl^3}{2E}\left(\frac{1}{I_1}+\sum_{n=2}^{6}\frac{n^3-(n-1)^3}{I_n}\right), \tag{5.119}$$

$$\textit{with}: \qquad I_n = \frac{bx_n^{\,3}}{12} \textit{ for } n = 1, 2, \ldots, 6, \tag{5.120}$$

$$E = \textit{Young modulus} = 2.06.10^5\ [N/mm^2]\ ; \tag{5.121}$$

- the multiple clutch brakes design (§ 5.4.2.7), where objective 2 is the stopping time t_h [s] (to be minimized).

To solve them, within the expert module, p_{rep} is chosen equal to 100%, since this value gave the best results for the single-objective counterparts of these three examples. Each process was launched 50 times, a whole process consisting in running the EA with both methods with a set of weights varying from $\{w_1^* = 0 ; w_2^* = 1\}$ for the first run to $\{w_1^* = 1 ; w_2^* = 0\}$ for the last run, by a constant step. Numerical results are gathered in Table 5.9.

Problem	***Number of feasible runs (over 50)***		***R1(PAMUC II,WS)***	
	Weighted sum method (with Joines and Houck)	*PAMUC II*	*Mean*	*Std. deviation*
Robot gripper design	*17*	*50*	*0.977*	*0.1415*
Beam design	*42*	*50*	*0.958*	*0.1559*
Multiple clutch brakes	*50*	*50*	*0.821*	*0.3283*

Table 5.9 : Comparison of PAMUC II and the weighted sum method for the 3-objective test cases.

Once again, those results confirm that PAMUC II clearly outperforms the weighted sum method (combined with Joines and Houck's method) : the simultaneous application of the expert module (to handle the rules) and of PAMUC (to tackle multicriteria and constrained aspects) is very effective.

Before drawing general remarks about PAMUC II (§ 5.4.6), two topics still have to be considered : the consistency of the rule base (§ 5.4.4) and the computational time (§ 5.4.5).

5.4.4 Consistency of the rule base

Some useful theorems have been mentioned in § 5.3.1.2, which guarantee that the expert module furnishes one (and only one) solution, in a finite time. Those theorems are applicable only if the rule base is logically consistent.

For example, if the following rules (see Eqs. (5.121) and (5.122)) are included in the rule base, their application would lead to a contradiction :

$$\begin{cases} \textit{rule 1} \equiv p \Leftrightarrow q & (5.122) \\ \textit{rule 2} \equiv p \Leftrightarrow \neg\, q & (5.123) \end{cases}$$

$$\rightarrow \textit{rules 1 and 2 : } p \Leftrightarrow \neg\, p \textit{ (contradiction !)} \qquad (5.124)$$

Lots of algorithms to check logical consistency of rule bases were developed in the logical programming field. The main one is Davis and Putnam's algorithm [LLO84], for which different implementations have been proposed [ZHA94]. In the frame of this work, it was implemented in Matlab (cf. flow-chart in pseudo-code in Figure 5.24). The first step consists in converting the rule base S^{rules} into a set S^{cl} of clauses, each clause being (by definition) a disjunction of a finite number of propositions (i.e. under the form : $p_1 \vee p_2 \vee \ldots \vee p_n$). To perform this transformation, the following rules are applied [THA90] :

$$\textit{1°) replace all } (X \Leftrightarrow Y) \textit{ by } (X \Rightarrow Y) \wedge (Y \Rightarrow X) \qquad (5.125)$$

$$\textit{2°) replace all } (X \Rightarrow Y) \textit{ by } (\neg\, X \vee Y) \qquad (5.126)$$

$$\textit{3°) use Morgan laws : } \neg\,(X \wedge Y) \textit{ is converted into } (\neg\, X \vee \neg\, Y) \qquad (5.127)$$

$$\neg\,(X \vee Y) \textit{ is converted into } (\neg\, X \wedge \neg\, Y) \qquad (5.128)$$

4°) finally, apply distributivity laws :

$$X \vee (Y \wedge Z) \textit{ is converted into } (X \vee Y) \wedge (X \vee Z) \qquad (5.129)$$

$$(X \wedge Y) \vee Z \textit{ is converted into } (X \vee Z) \wedge (Y \vee Z) \qquad (5.130)$$

Then, a recursive procedure is applied, where the verification of S^{cl} consistency is replaced by the checking of the consistency of two smaller subsets which do not contain a proposition p (resp. $\neg\, p$).

The rule bases of test cases TCR 1 to TCR 7 have been checked by Davis and Putnam's algorithm, which proved that all of them were logically consistent.

```
begin
    transform the original rule base $S^{rules}$ into a set of clauses $S^{cl}$
    if rule base $S = \{\ \}$ then
        $S$ is consistent
    elseif $S = \{ False \}$ then
        $S$ is inconsistent
    else
        - select a proposition $p$ intervening in $S$
        - calculate $S_p$, $S_{\neg p}$ and $S'' = S \setminus (S_p \cup S_{\neg p})$
        - calculate $S_p'$ (whose clauses are clauses of $S_p$ without $p$)
        - calculate $S_{\neg p}'$ (whose clauses are clauses of $S_{\neg p}$ without $\neg p$)
        - $S$ is inconsistent iff both $(S_p' \cup S'')$ and $(S_{\neg p}' \cup S'')$ are
          inconsistent
    end
end
```

Fig. 5.24 : Flow-chart of Davis and Putnam's algorithm (adapted from [LLO84]).

Nevertheless, one should be careful about the fact that the content of the propositions constituting the rules does not intervene in Davis and Putnam's algorithm : only logical expressions are taken into account. For example, if propositions *p, q* and *r* are defined as follows :

$$p \equiv x_1 = 3, \tag{5.131}$$
$$q \equiv x_2 = 5, \tag{5.132}$$
$$r \equiv x_1 = 6, \tag{5.133}$$

and if the rule base contains :

$$rule\ 1 \equiv p \Rightarrow q, \tag{5.134}$$
$$rule\ 2 \equiv q \Rightarrow r, \tag{5.135}$$

the inference engine will deduce that $p \Rightarrow r$ (i.e. $(x_1 = 3) \Rightarrow (x_1 = 6)$), which is logically consistent with the rule base but mathematically false. To check at once logical and "mathematical" consistency, a *constraint satisfaction program* should be solved.

Constraint satisfaction programming (CSP) aims at predicting, from a set of constraints, whether the admissible domain (i.e. the set of points respecting all the constraints) either exists or is empty [BOW90]. While efficient algorithms were developed for specific cases (linear constraints for instance [HOO02]), CSP with rules, handling general mathematical expressions (as it is often the case in design optimization, e.g. with empirical formulas), is still an open area. By the way, it is interesting to notice that genetic algorithms, thanks to their robustness, have been used to solve this kind of problem (cf. [LAU99]).

However, as the scope of PAMUC II is to tackle pre-design optimization problems, with a quite low number of rules (≤ 50), one can reasonably assume that the user can himself/herself detect any "mathematical" contradiction in the rule base ; hence only a verification of the consistency is performed numerically.

5.4.5 Computational time

The computational time of PAMUC II is investigated in this section. A decomposition of the different costs leads to the following formula :

$$T_{1gen}^{PAMUC} = N.[T_{sel\_PROM\_II} + T_{cross} + T_{mut} + T_{obj} + T_{constr} + N_{rules}{}^2.p_{rep} + (1 - p_{rep}).N_{rules} + (N+1).(m+1).T_{PROM\_II} + K''], \qquad (5.136)$$

where :

- $T_{1gen}^{PAMUC\ II}$ is the computational time needed for one generation of the EA with PAMUC II ;
- $T_{sel\_PROM\_II}$ is the average time (for one member of the population) to perform selection (by an elitist selection procedure using PROMETHEE II for each pair of parents and their corresponding children) ;
- m is the number of objective functions ;
- N_{rules} is the number of rules ;
- p_{rep} is the probability of replacement ;
- T_{PROM_II} is the time needed to compute (for a couple of individuals (a,b)) the preference functions $P_i(a,b)$ and the preference indexes $\pi(a,b)$ needed to rank the individuals in PROMETHEE II ;
- K'' is a second-order term for remaining (low cost) computations of the algorithm.

With a classical weighted sum method (combined with Joines and Houck’s penalty technique to handle both mathematical constraints and expert rules), the computational cost for one generation is equal to :

$$T_{1gen}^{WS} = N.[T_{sel} + T_{cross} + T_{mut} + T_{obj} + T_{constr+rules} + K^*], \qquad (5.137)$$

where :

- T_{1gen}^{WS} is the computational time needed for one generation of the EA with the weighted sum method ;
- N is the size of the population ;
- T_{sel} is the average time (for one member of the population) to perform selection ;
- T_{cross} is the average time (for one member of the population) to perform crossover ;
- T_{mut} is the average time (for one member of the population) to perform mutation ;
- T_{obj} is the time needed to compute the values of the m objective functions (for one individual) ;
- $T_{constr+rules}$ is the time needed to compute (for one individual) the values of the $p+q$ (equality and inequality) constraints, as well as the N_{rules} converted into mathematical constraints ;
- K^* is a second-order term for remaining (low cost) computations of the algorithm.

Whilst the weighted sum method performs faster than PAMUC II, it was shown above that it often has hindrances to find a feasible solution, even with a rather low number of rules. Furthermore, the theoretical results mentioned in Eqs. (5.135) and (5.136) focus on the computation of a single generation, whilst convergence towards the optimum can be reached in less generations with PAMUC II, hence diminishing the total calculation time.

First, a deeper insight will be made about the cost due to the expert module in PAMUC II. Therefore, a computational time analysis (enlightening the role of the probability of replacement) is done on test cases TCR 1 and TCR 2 (with the same EA parameters as in §§ 5.4.2.1 and 5.4.2.2), beginning with a closer look at the number of generations. Figures 5.25 and 5.26 exhibit the maximum number of generations that can be computed for a given computational time *CT*. That means that in test case TCR 1, when 15 generations can be performed for $p_{rep} = 0$, only 13 generations can be accomplished during the same time.

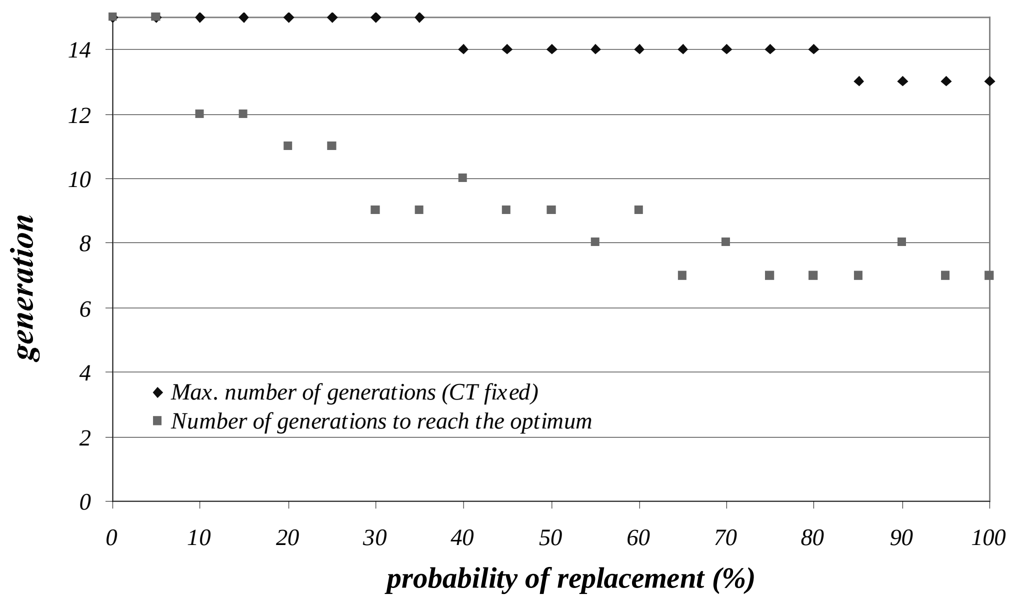


Fig. 5.25 : Computational time study for TCR 1 : diamonds represent the maximum number of generations that can be computed in a given computational time CT (it logically decreases when p_{rep} increases), whereas squares depict the actual number of generations needed to reach the global optimum $\mathbf{x}^*$.

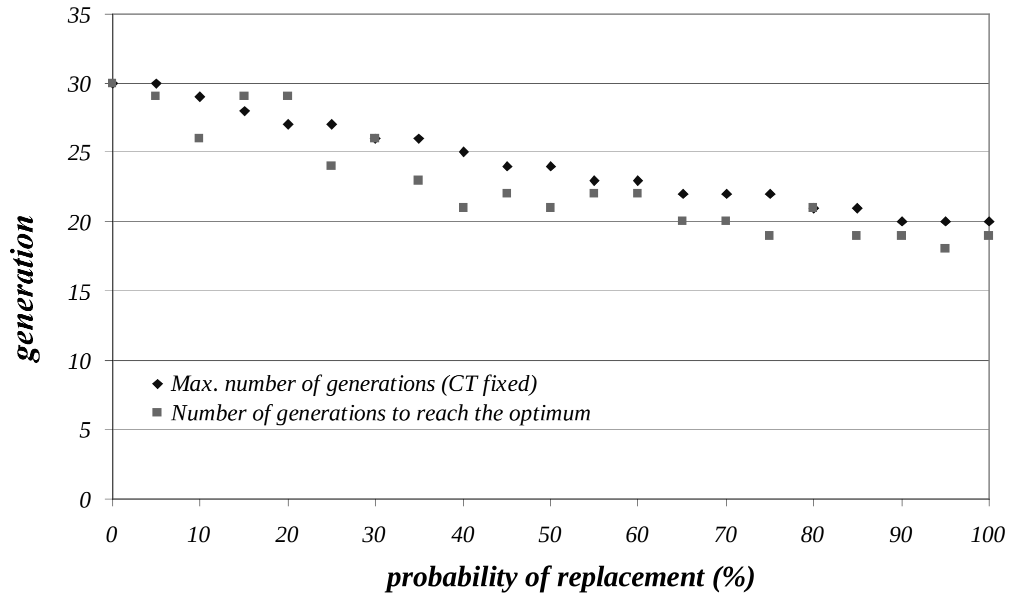


Fig. 5.26 : Computational time study for TCR 2 : diamonds represent the maximum number of generations that can be computed in a given computational time CT (it logically decreases when p_{rep} increases), whereas squares depict the actual number of generations needed to reach the global optimum $\mathbf{x}^*$.

One can also see that the number of generations needed to reach the optimum decreases (approximatively) linearly w.r.t. p_{rep} : in TCR 1, while 30 generations are necessary to reach the optimum with $p_{rep} = 0$, only 19 are sufficient to find it when $p_{rep} = 100\%$. But as the calculation cost of one generation increases with p_{rep}, one must compare directly the CPU times spent to find $\boldsymbol{x}^*$: this is exhibited in Fig. 5.27, showing that in TCR 1, the cost decreases w.r.t. p_{rep}, whereas no correlation can be made in TCR 2.

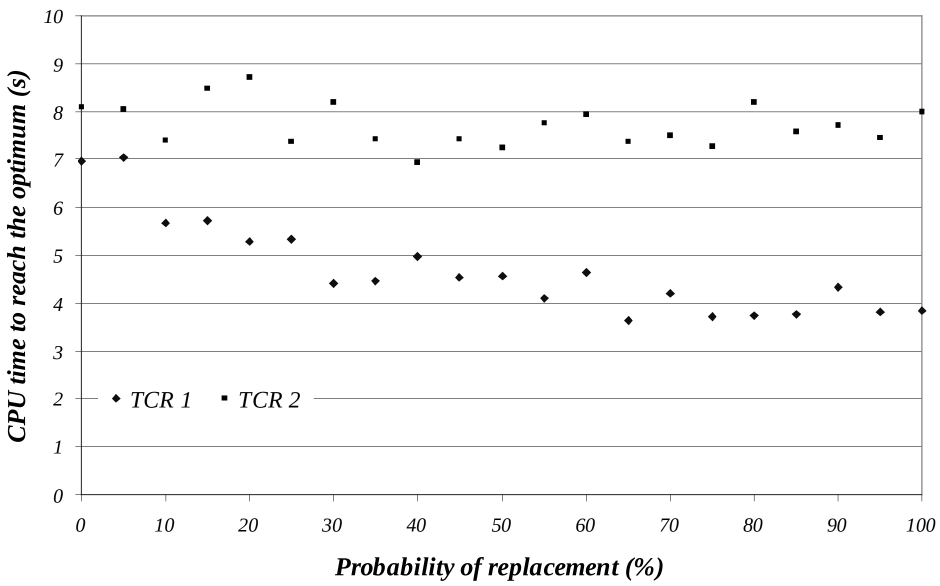


Fig. 5.27 : CPU time needed to reach the optimum with PAMUC II w.r.t. p_{rep} (for test cases TCR 1 and TCR 2) .

Now PAMUC II and the weighted sum method are compared following the CPU time needed for one run of the EA (with the same EA parameters as in § 5.4.2, and $p_{rep} = 100\%$ in the expert module ; results were obtained on a PC with *freq.* = 900 MHz in MS Windows 2000 environment). Numerical results for TCR 3 to TCR 7 are indicated in Table 5.10. One should insist on the fact that although PAMUC II generates an additional time, it is still very reasonable since it furnishes much better results in terms of admissibility and quality of the solution (cf. results in § 5.4.2).

Problem	***Name of the problem***	N_{rules}	N_{constr}	***CPU time (s)***		
				Joines and Houck	***PAMUC II***	***Relative difference (%)***
TCR 3	*Hooker et al.*	*4*	*1*	*0.501*	*0.521*	*3.8%*
TCR 4	*Robot gripper*	*3*	*6*	*238.2*	*240.6*	*1.0%*
TCR 5	*Beam design*	*2*	*11*	*58.5*	*76.5*	*23.5%*
TCR 6	*Helical spring*	*4*	*7*	*61.1*	*66.3*	*7.8%*
TCR 7	*Multiple clutch brakes*	*3*	*9*	*23.2*	*30.1*	*22.9%*

Table 5.10 : CPU time (for one run of the EA) for test cases TCR 3 to TCR 7 (the high CPU time for TCR 4 is due to the fact that Nelder and Mead's algorithm is utilized twice for each individual in order to compute its objective function : cf. § 5.4.2.4).

5.4.6 Remarks about PAMUC II

The development of a novel optimization method, like PAMUC II, is traditionally divided in three parts : first, the bibliographical study of the field concerned, followed by the theoretical justification of all the options taken when devising the method. Then, the proposed approach must be tested on variegated benchmarks, in order to validate it and surround its possible deficiencies. And finally comes the drawing of the conclusions, to derive the advantages and caveats of the method. It is the scope of this section.

First, the satisfactory results obtained by the expert module in PAMUC II can be explained thanks to an original concept taken from CSP, namely *epistasis*, inspired by genetics, and can be defined as “the interaction between different genes in a chromosome” [LAU99]. In other words, as standard EAs act as black-box and do not use any specific knowledge about the problems they try to solve, they have hindrances in applications where there are interwoven relationships between the variables, which are difficult to detect by a mere blind search [TRO97]. In those problems, the harmful effects of epistasis may be alleviated by incorporating additional knowledge into the algorithm. This explains why PAMUC II succeeds in converging towards the optimal solution even when the size of the admissible domain is narrow in comparison with the whole search space.

The other advantages of PAMUC II are the following :

- new rules may be added to the rule base without changing the structure of the algorithm, as long as the base of rules is not contradictory. The verification of the base consistency has been discussed in § 5.4.4, leading to the implementation of Davis and Putnam’s algorithm to check the logical consistency. However, “mathematical” consistency is not automatically analyzed, and must be verified by the user. For example, it should be noticed that no loops or retroactivity in the rules are tolerated, since it would lead to misleading results (the rule base must always be transformed into an *AND-OR* tree as exposed in § 5.2.1.3) ;

- as the language used to model the rules is based upon propositional calculus, a unique solution is guaranteed to be found each time the expert module is launched ;

- the discussion about the value to assign to the new parameter p_{rep} led to the conclusion that p_{rep} = 100% gave the best results, also because the problem with rules are often epistatic (i.e. there are relationships between the variables). However, when a new problem is tested, a tuning of p_{rep} should always be performed, because giving a too large value to p_{rep} may accelerate the convergence to a small part of the search space, without letting time enough to the EA to explore the whole domain ;

- the study of computational cost showed that PAMUC II needs only a minor additional time compared to the cheap weighted sum method combined to Joines and Houck’s technique (less than 50% for all the examples treated in this work) ;

- the implementation of PAMUC II into the standard EA is quite easy, even if the writing of the rule requires the user to be able to change the chromosome of the members of the population, hence to act on the coding.

In addition to those remarks related to the expert module, it can be underlined that the interaction of PAMUC (to deal with multiple objectives and mathematical constraints) and the expert module works efficiently, whence all the advantages of PAMUC can be reminded, namely :

- several objectives with different scales of sizes may be used ;
- no tuning of parameters is required (except p_{rep} as discussed above, and the traditional parameters of the EA as the size of the population N, etc.) ;
- it was able to find solutions distributed along concave Pareto fronts (by varying the weights), whereas the weighted sum method was attracted next to extreme parts of the trade-off surface. However, the possible user of PAMUC II must be aware that this property may not appear systematically ;
- both PAMUC and PAMUC II have proven to be robust and efficient on a large set of test cases and mechanical design optimization problems, and the validation was made by a rigorous use of *R1*-norm for multiobjective problems. Industrial applications will be handled in the next chapter.

5.5 Conclusions

This chapter was devoted to the use of expert rules within the optimization process, in order to generate optimal solutions satisfying also technological (or other) requirements. A preliminary bibliographic survey about expert systems used in engineering applications showed that their lack of flexibility makes them difficult to use as a pre-design optimization tool.

However, adding knowledge in EAs is an elegant way to guide the algorithm towards the feasible global optimum, instead of letting it groping for the admissible domain. Therefore, a novel approach was proposed, called PAMUC II, consisting in modelling expert rules thanks to the propositional calculus, and using them (with a user-defined probability of replacement p_{rep}) to repair the members of the population violating those rules.

It was validated for various single-objective and multiobjective benchmarks (in which supplementary rules were introduced in some cases in order to increase the complexity of the problems), and PAMUC II gave excellent results in comparison with a weighted sum method (with a dynamic penalty technique to tackle the constraints).

Furthermore, Davis and Putnam’s algorithm was implemented to check the logical consistency of rule bases ; mathematical consistency is not investigated since it requires as much computational overhead as the optimization process. In any case, for most pre-design optimization problems, the number of rules is generally restricted to small or medium values (≤ 50), thus the user can check by himself/herself whether there is a contradiction in the rule base.

Then, a computational time study exhibited a slight difference between PAMUC II and a classical weighted sum – penalty method, which makes PAMUC II very effective, robust and rather cheap.

Finally, some remarks have underlined the role of the probability of replacement, whose optimal value in the test cases cited above is equal to 100%, but which must be tuned for every new example in order to be sure that the EA will not be trapped in the vicinity of a local optimum.

Now that PAMUC II has been thoroughly validated and its behaviour dissected, its application on industrial designs, which has been hived off from the last three chapters, will be presented in the next chapter.

Chapter 6 – Industrial Applications

This chapter – concerning Techspace Aero (Snecma group) valves – is strictly confidential.

CHAPTER 7 – CONCLUSIONS

7.1 Pre-design optimization with expert rules

The goal of this thesis was to contrive an efficient and robust method to perform the optimization of mechanical components during the first stage of the design process. This is a crucial subject, since a minor modification in the pre-design may bring significant improvements to the final structure, whilst an “ill” design, even efficiently optimized at the very end of the dimensioning, will perform poorly. However, most algorithmic methods in structural optimization are concerned with the last stage of the design process. This is partly due to the fact that in industrial contexts, for complex mechanical components (as valves, pomps, etc.) already characterized by several parts (springs, piston, bellows, screw bolts, etc.), engineers choose the best preliminary design among different configurations, not only with respect to quantitative requirements of performance or cost, but also by allowing for technological considerations.

Those latter requirements are more easily translated in terms of logical rules (≡ *IF* condition *THEN* conclusion), like in expert systems. These algorithms use knowledge collected amid experts in a specific field to work out complex problems. Nevertheless, the building of an expert system is a tedious task, and its use is confined to very restricted applications. Thence, the scope of this work was to set forth a method able to optimize mechanical components for a wide range of applications, but by taking expert knowledge into account.

To accomplish this task, the development of the optimization method was divided in two steps : first, one should be able to optimize parameterized designs, for which the topological configuration is fixed and only geometrical and material variables are involved. Then, the method should be extended to more general applications (with topological variables), by handling also expert rules. This division in two steps is related to the simple statement that it is pointless to compare an intrinsically “poor” design whose geometry would have been correctly optimized to a “better” design not parametrically optimized. Of course, it should be emphasized that this separation is more particularly adapted to components endowed with a certain complexity (as valves, compressors, turbines for instance), and less to simpler mechanical parts (like bolts, seals, etc.) which can be sketched “from scratch” and directly compared without undergoing an optimization step.

7.2 PAMUC for parametrical design optimization

Once a design has been parameterized, its most significative geometrical and material *variables* must be determined by the user, whereafter technical requirements are considered (the *constraints*), as well as the *objective* function(s). As soon as the problem is formulated mathematically, an appropriate algorithm has to be chosen among the huge amount of methods proposed in the literature. They branch off in two categories : local methods, generally based on the computation of the sensitivities, and global methods, incorporating metaheuristics, whose most widespread instances are evolutionary algorithms (EAs). They are based on the Darwinian model of “survival of the fittest”, wherein the best members of a population of potential solutions are favoured and combined in order to create better individuals at the next generation.

EAs need only the values of the functions, and not their derivatives, which makes them very attractive for design optimization, which deals as often as not with mixed variables and non differentiable functions, banishing *de facto* the use of gradient-based algorithms. They have proven to furnish remarkable results for single-objective unconstrained problems ; nevertheless, it is also well-known that they have hindrances to tackle constraints. Furthermore, in industrial context, more than one objective have to be considered.

To understand multiobjective optimization, the concept of Pareto solution was introduced, defining $\boldsymbol{x}^*$ as a nondominated vector if and only if there exists no other $\boldsymbol{x}$ in the feasible domain such that $\forall\ i \in \{1,\ldots,m\}$, $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*)$ and for at least one $i \in \{1,\ldots, m\}$: $f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^*)$ (for a minimization problem). This fundamental definition led to separate the multiobjective methods in three approaches :

- in *a posteriori methods*, preferences are used at the end, when the Pareto front (i.e. the image in the objective space of the nondominated vectors) has been completely determined ;
- in *progressive methods*, preferences are used during the optimization process, in an interactive way ;
- in *a priori methods*, the decision maker's preferences about the objectives are expressed before the search process, in terms of weights or a ranking.

In this work, an a priori approach was chosen. Indeed, so far, most researchers focus on a posteriori techniques, while – as Coello underscored it – few studies are explicitely concerned with the simultaneous handling of the preferences and the constraints within EAs, though it can be a suitable tool for pre-design optimization. Therefore, a novel method was proposed for parametrical optimization : PAMUC (*Preferences Applied to MUltiobjectivity and Constraints*). Its main features are the following : after having considered the satisfaction of the constraints as a m+1th objective (m being the number of objectives) , all individuals of the population are selected following those m+1 criteria thanks to an outranking method, PROMETHEE II, developed by Brans and Mareschal in the multicriteria decision aid field. Moreover, adaptive weights are used to compel the population to progressively converge towards the feasible domain : the weight $w_{m+1}^{(t)}$ assigned to the m+1th objective is proportional to the number of admissible individuals at current generation t, and the other weights are computed in such a way that the relative proportion between them (initially defined by the user) are preserved, and that all weights add up to 1.

The validation of PAMUC was performed in two stages : first, single-objective constrained problems have been analyzed, in order to check the efficiency of the method in tackling constraints. Numerical results showed that PAMUC gives very satisfactory solutions in comparison with constraint-handling techniques specially devised for 1-objective problems.

Then, a special procedure had to be selected to validate PAMUC on multiobjective applications. As the most widespread approach in industrial context is the linear aggregation of the criteria, a set of standard test cases were applied to PAMUC as well as the classical weighted sum (combined with a dynamic penalty technique to deal with the constraints). The key idea of the validation was thus to compare the nondominated solutions obtained by both methods for different values of the weights.

After a thorough discussion about the different indicators exposed in the literature to compare multiobjective sets, propping itself against the theoretical works of Knowles and Corne, and

Zitzler *et al.*, it resulted that the most appropriate norm to compare two sets was *R1*-norm, proposed by Hansen and Jaszkiewicz. It consists in using a set of utility functions (here : weighted Tchebycheff functions) and defining an indicator describing the outperformance of one set over another with respect to each utility function. Then, an overall norm *R1* is computed, by integrating the indicators over the whole set of utility functions. Multiobjective test cases (with N_{var} = number of variables ≤ 20 and N_{constr} = number of constraints ≤ 50) were analyzed hereby, and showed very clearly that PAMUC outperforms the traditional penalized weighted sum method, to reach the feasible domain and to find optimal solutions.

One of the most interesting advantage of PAMUC is that no tuning of parameters is required. Indeed, the only parameters related to PAMUC are the preference and the indifference indexes from PROMETHEE II. Normally they should be defined by the user for each criterion. However, when this information is not available, values of 1 for the preference indexes p_i and of 0 for the indifference indexes q_i gave very satisfactory results in all the applications mentioned in this thesis. Furthermore, the method has demonstrated to be robust with respect to small variations of the weights. Finally, it is also important to underline that this enhancement of the results obtained thanks to PAMUC needs only a reasonable supplementary amount of time in comparison with the weighted sum method (less than 50% in the most unfavourable example treated in this work).

A caveat which a potential user of PAMUC should pay heed to is that some multiobjective are hard to solve, i.e. the nondominated points found with different values of the weights may be located in the extreme sides of the trade-off surface, instead of being distributed uniformly all along it. In those cases, an a posteriori method must be used first in order to locate the Pareto front, and then a multicriteria decision aid method applied to the nondominated points found during the search process.

Once the methodology was validated, it has been implemented in Boss Quattro (Samtech s.a.), a commercial software for parametrical studies and optimization. In this environment, PAMUC was used for the parametrical optimization of two industrial applications, namely two poppet valves designed by Techspace Aero (Snecma group) for the VINCI engine from launcher Ariane 5. The first application was characterized by strong technical requirements that restricted drastically the size of the admissible space. In this case, whereas two constraint-handling techniques (a dynamic penalty technique and a method separating feasible from unfeasible solutions) were inefficient to find admissible solutions, PAMUC bore out to the assertion that it was a robust tool since it found a solution satisfying all the constraints.

The study of the second valve, dealing with two objectives (minimizing the volume whilst maximizing the performance), also showed better results for PAMUC in comparison with a traditional weighted sum method (combined with a dynamic penalization to tackle the constraints).

7.3 PAMUC II for design optimization with expert rules

As soon as the first step was achieved, the development of the second step might begin, namely incorporating expert rules within the optimization process, in order to optimize more general models with topological variables and having to respect technological (or other) requirements.

Expert knowledge can often be interpreted as logical rules. When an important amount of information has been collected among experts, specific algorithms can be developed to manipulate the knowledge and solve specific problems : *expert systems*. However, a preliminary bibliographic survey about expert systems used in engineering applications illustrated their lack of flexibility, which makes them ill adapted for pre-design optimization.

Nevertheless, adding knowledge to EAs is well adapted to guide the algorithm towards the feasible global optimum, instead of letting it seeking "blindly" the admissible domain, as shown by Michalewicz. This is also confirmed by the No Free Lunch theorem, introduced by Wolpert and Macready, which states that there exists no general "black-box" algorithm able to give optimal solutions for all families of problems ; it means that solving efficiently specific problems requires to add knowledge within the optimization procedure.

Therefore, a novel approach was proposed, called PAMUC II, extending the multicriteria optimization method developed during the first step. It consists in modelling expert rules thanks to the propositional calculus, and using them (with a user-defined probability of replacement p_{rep}) to repair the members of the population violating those rules.

PAMUC II was validated on a set of standard single-objective and multiobjective benchmarks, either purely "mathematical" or mechanical (in which supplementary rules have been introduced in some cases in order to increase the complexity of the problems), and gave excellent results in comparison with a weighted sum method (with a dynamic penalty technique to tackle both the constraints and the rules). Moreover, PAMUC II has the following advantages :

- new rules may be added to the rule base without changing the structure of the algorithm, as long as the rule base is not contradictory. It should be noticed that no loops or retroactivity in the rules is tolerated, since it would lead to misleading results ;

- as the language used to model the rules is based upon propositional calculus, a unique solution is guaranteed to be found each time the expert module is launched ;

- the discussion about the value to assign to the new parameter p_{rep} led to the conclusion that p_{rep} = 100% gave the best results. However, when a new problem is tested, a tuning of p_{rep} should always be performed, because assigning a too large value to p_{rep} may accelerate the convergence to a small part of the search space, without letting time enough to the EA to explore the whole domain ;

- the computational cost study showed that PAMUC II needs only a minor additional time compared to the weighted sum method combined to Joines and Houck's technique (less than 50% for all the examples treated in this work) ;

- the implementation of PAMUC II into the standard EA is quite easy, even if the writing of the rule requires the user to be able to change the chromosome of the members of the population, hence to act on the coding.

Furthermore, Davis and Putnam's algorithm was implemented to check the logical consistency of rule bases. "Mathematical" consistency – dealing with the content of the propositions – is not investigated in the frame of this thesis, because the possible presence of complex expressions in the propositons implicate that no general algorithm can be built to verify systematically the rule base consistency : a constraint satisfaction program is to be solved for each particular rule base. As in most pre-design optimization problems the number of rules is generally restricted to quite small values (≤ 50), it is assumed that the user can check immediately the consistency of the rule base.

The last task to achieve was to confront PAMUC II with industrial applications, namely the valves mentioned above, and a third one, which are all poppet valves designed by Techspace Aero (TA – Snecma group) for the VINCI engine from launcher Ariane 5. The first step was to collect information among TA engineers about the methodology of design and dimensioning of poppet valves. The corresponding knowledge was translated in terms of expert rules, directly written as logical expressions. Then, a general poppet valve model was built, synthetizing different configurations of valves by means of topological variables. Finally, PAMUC II was applied and compared to a penalized weighted sum method. Numerical results clearly demonstrated that for all valves, PAMUC II outperformed the penalized weighted sum technique, by furnishing optimal solutions satisfying technical requirements as well as technological constraints.

It follows therefrom that PAMUC II has demonstrated on many examples to be an efficient tool for pre-design optimization of mechanical components. Future prospects could take tolerances of the variables into account, for example by means of fuzzy rules, or by statistical variables.

Another possible future development of this work could be to study PAMUC II behaviour in other fields of engineering, since no physical hypothesis was assumed in the algorithm. Furthermore, in order to handle more complex representation of the knowledge, e.g. with contradictory rules (for which the user would be able to rank the relative importance of the rules, indicating that rule 1 must be applied in priority w.r.t. rule 2 for instance), a higher level logical language could be used. However, it should be underscored that it would involve a risk of non convergence and a higher computational time, whereas propositional calculus – as used in PAMUC II – is often sufficiently rich to model complex knowledge.

7.4 Original contributions of the thesis

The main original contributions of this thesis are summarized hereafter :

- a novel multicriteria optimization method (PAMUC) is proposed for parameterized pre-designs, considering the satisfaction of the constraints as a new objective and using a multicriteria decision aid method (PROMETHEE II) to rank the objectives. The use of adaptive weights (related to the rate of feasible individuals at each generation) seems efficient to obtain admissible results, whilst the outranking performed by PROMETHEE II furnishes better solutions than a traditional – albeit widespread in industrial context – weighted sum method ;

- numerical results obtained by PAMUC and the weighted sum method are preceded by a thorough discussion about the indicator to be used to compare both methods ; to the author's knowledge, it is the first time that this rigorous approach (i.e. finding nondominated solutions for different values of the weights and using Hansen and Jaszkiewicz's *R1*-norm to compare them) was applied to validate an *a priori* method (i.e. using the user's preferences since the very start of the seach process) ;

- the second version of PAMUC, incorporating logical rules in the design process, is an original approach to heed expert knowledge. PAMUC II integrates an inference engine within the EA, whose purpose is to verify whether each individual (with a user-defined probability) satisfies some expert rules (e.g. representing technological requirements) ; if not, it is repaired according to them. Test cases, mechanical benchmarks and industrial applications all show that it gives very satisfactory results in comparison with a "blind" search handling the constraints solely by penalization, and it is also worth mentioning that it needs only a reasonable additional computational time ;

- for the industrial applications, namely the valves designed by Techspace Aero (TA) for launcher Ariane 5, a general poppet valve model was built, synthetizing different possible configurations. Furthermore, expert rules concerning the valve design were collected among TA engineers, and the quality of PAMUC II results obtained on three valves confirmed the benefits of using it as an automation and optimization tool for pre-designs.

REFERENCES

References are classified by alphabetical order (of the first author).

[ABE96] B. Aberšek, J. Flašker & J. Balič, Expert System for Designing and Manufacturing of a Gear Box, Expert Systems with Applications, vol. 11 (3), pp. 397-405 (1996).

[AFO02] S.M.B. Afonso, C.M.H. Macedo & D.A.P. Oliveira, Structural Shape Optimization Under Multi-criteria Conditions, WCCM V – Fifth World Congress on Computational Mechanics, July 7-12 (2002).

[AGU03] A.H. Aguirre, S.B. Rionda, G.L. Lizárraga, C.A.C. Coello, IS-PAES: A Constraint-Handling Technique Based on Multiobjective Optimization Concepts, Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), pp. 73-87 (2003).

[ALO01] P. Alotto & M.A. Nervi, An efficient hybrid algorithm for the optimization of problems with several local minima, International Journal for Numerical Methods in Engineering, vol. 50, pp. 847-868 (2001).

[AND00] J. Andersson, A Survey of Multiobjective Optimization in Engineering Design, Technical report LiTH-IKP-R-1097, Department of Mechanical Engineering, Linköping University, Linköping, Sweden, 34 pp. (2000).

[ANI02] W. Annicchiarico & M. Cerrolaza, Comparison of Binary and Real Coded Genetic Algorithms in the Solution of Difficult Shape Optimization Problems, WCCM V – Fifth World Congress on Computational Mechanics, July 7-12 (2002).

[AZI02] I.A. Azid, A.S.K. Kwan & K.N.Seetharamu, A GA-based technique for layout optimization of truss with stress and displacement constraints, International Journal for Numerical Methods in Engineering, vol. 53, pp. 1641-1674 (2002).

[BAC91] Th. Bäck, F.H. Hoffmeister & H.-P. Schwefel, A Survey of Evolution Strategies, Proceedings of the 4th International Conference on Genetic Algorithms, pp. 2-9 (1991).

[BAC92] Th. Bäck, Evolutionary Algorithms, ACM SIGBIO Newsletter, pp. 26-31 (1992).

[BAC93] Th. Bäck, G. Rudolph & H.-P. Schwefel, Evolutionary Programming and Evolution Strategies : Similarities and Differences, Proceedings of the Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, San Diego CA pp. 11-22 (1993).

[BAC96] Th. Bäck & H.-P. Schwefel, Evolutionary Computation : An Overview, Proceedings of 1996 IEEE International Conference on Evolutionary Computation (ICEC '96), May 20-22, 1996, Nayoya University, pp. 20-29 (1996).

[BAC97] Th. Bäck, D.B. Fogel & Z. Michalewicz (eds.), Handbook of Evolutionary Computation, Oxford University Press, New York, 988 pp.(1997).

[BAL01] R. Balling & S. Wilson, The Maximum Fitness Function for Multi-objective Evolutionary Computation: Application to City Planning, in L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M.H. Garzon & E. Burke (eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 1079-1084, Morgan Kaufmann Publishers, San Francisco, California, July (2001).

[BEC00] M. Beckers, Dual methods for discrete structural optimization problems, International Journal for Numerical Methods in Engineering, vol. 48, pp. 1761-1784 (2000).

[BLA98] J.A. Bland, A memory-based technique for optimal structural design, Engineering Applications of Artificial Intelligence, vol. 11 (3), pp. 319-325 (1998).

[BOS01] BOSS-Quattro v4.1, Manuel d'utilisation, Samtech S.A. 8, rue des Chasseurs Ardennais, 4031 Angleur (2001).

[BOW90] J. Bowen & D. Bahler, Constraint Processing and Logic Programming, AAAI Workshop on Constraint Directed Reasoning, July (1990).

[BRA84] V. Braibant & Cl. Fleury, Shape optimal design using B-splines, Computer Methods in Applied Mechanics and Engineering, vol. 44, pp. 247-267 (1984).

[BRA86] J.-P. Brans & B. Mareschal, How to select and how to rank projects : The PROMETHEE method for MCDM, European Journal of Operational Research, vol. 24, pp. 228-238 (1986).

[BRA94] J.-P. Brans & B. Mareschal, The PROMCALC and GAIA decision support system for Multicriteria decision aid, Decision Support Systems, North Holland, vol. 12, pp. 297-310 (1994).

[BRA96] J.-P. Brans, The space of freedom of the decision maker – Modelling the human brain, European Journal of Operational Research, vol. 92, pp. 593-602 (1996).

[BRE98] G. Brewka & J. Dix, Knowledge Representation with Logic Programs, Logic Programming and Knowledge Representation, Springer LNAI 1471, p. 1-55 (1998).

[BUR02] T. Burczyński & P. Orantek, Evolutionary Algorithms in Computational Mechanics : Applications in Optimization and Identification, WCCM V – Fifth World Congress on Computational Mechanics, July 7-12 (2002).

[CAM97] E. Camponogara & S.N. Talukdar, A Genetic Algorithm for Constrained and Multiobjective Optimization, 3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA), J.T. Alander, editor, Vaasa, Finland, University of Vaasa, pp. 49-62 (1997).

[CAO00] Y.J. Cao, L. Jiang & Q.H. Wu, An evolutionary programming approach to mixed-variable optimization problems, Applied Mathematical Modelling, vol. 24 (12), pp. 931-942 (2000).

[CHA85] E. Charniak & D. McDermott, Introduction to Artificial Intelligence, Addison-Wesley Series in Computer Science, Reading, Massachusetts, USA, 699 pp. (1985).

[CHA03] K.W. Chau & F. Albermani, A coupled knowledge-based expert system for design of liquid-retaining structures, Automation in Construction, Vol. 12 (5), pp. 589-602 (2003).

[CIA98] Ph.G. Ciarlet, Introduction à l'analyse numérique matricielle et à l'optimisation, Editions Dunod, Paris (1998).

[COE96] C.A.C. Coello, An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design, Thèse de doctorat, Department of Computer Science, Tulane University, Mexique (1996).

[COE99] C.A.C. Coello, A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems, An International Journal, 1(3), pp. 269-308 (1999).

[COE99a] C.A.C. Coello, A Survey of Constraint Handling Techniques used with Evolutionary Algorithms, Technical Report Lania-RI-99-04, Laboratorio Nacional de Informática Avanzada (1999).

[COE00a] C.A.C Coello, Use of self-adaptive penalty approach for engineering optimization problems, Computers in Industry 41, pp. 113-127 (2000).

[COE00b] C.A.C. Coello, Handling Preferences in Evolutionary Multiobjective Optimization : A Survey, 2000 Congress on Evolutionary Computation, volume 1, pages 30-37, Piscataway, New Jersey (2000).

[COE00c] C.A.C. Coello, Treating Constraints as Objectives for Single-Objective Evolutionary Optimization, Engineering Optimization, vol. 32 (3), pp. 275-308 (2000).

[COE00d] C.A.C. Coello, A Micro-Genetic Algorithm for Multi-Objective Optimization, Lania-RI-2000-06, Laboratorio Nacional de Informática Avanzada (2000).

[COE02] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms : a survey of the state of the art, Computer Methods in Applied Mechanics and Engineering, 191, pp. 1245-1287 (2002).

[COE02a] C.A.C. Coello, D.A. Van Veldhuizen & G.B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic/Plenum Publishers, New York, 576 pp. (2002).

[COE02c] C.A.C Coello & E. Mezura-Montes, Handling Constraints in Genetic Algorithms Using Dominance-Based Tournaments, In I.C. Parmee (ed.), Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM) (2002).

[COL02] Y. Collette & P. Siarry, Optimisation multiobjectif, Eyrolles ed., Paris, 322 pp. (2002) .

[COR03] D.W. Corne & J.D. Knowles, No Free Lunch and Free Leftovers Theorems for Multiobjective Optimization Problems, Evolutionary Multi-Criterion Optimization (EMO 2003), Second International Conference, Faro, Portugal, Proceedings, pp. 327-341 (2003).

[COS01] L. Costa & P. Oliveira, Evolutionary algorithms approcah to the solution of mixed integer non-linear programming problems, Computers and Chemical Engineering 25, pp. 257-266 (2001).

[CRA01] G.W. Craenen, A.E. Eiben & E. Marchiori, How to Handle Constraints with Evolutionary Algorithms, Practical Handbook of Genetic Algorithms, Second Edition, L. Chamber ed., Chapmann & Hall/CRC Press, Ch 10, pp. 341-361 (2001).

[CVE99] D. Cvetković & I.C. Parmee, Use of preferences for GA-based multi-objective optimization, GECCO-99 : Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 1504-1509, Orlando, Florida, USA (1999).

[CVE00] D. Cvetković, Evolutionary Multi-Objective Decision Support Systems for Conceptual Design, PhD Thesis, School of Computing, Faculty of Technology, University of Plymouth (2000).

[CVE02] D. Cvetković & I.C. Parmee, Preferences and their Application in Evolutionary Multiobjective Optimisation, IEEE Transactions on Evolutionary Computation, vol. 6 (1), pp. 42-57 (2002).

[DEB95] K. Deb & R.B. Agrawal, Simulated binary crosssover for continuous search space, Complex Systems vol. 9, pp. 115-148 (1995).

[DEB96] K. Deb & M. Goyal, A combined genetic adaptive search (GeneAS) for engineering design, Computer Science and Informatics vol. 26 (4), pp. 30-45 (1996).

[DEB98] K. Deb & M. Goyal, A robust optimization procedure for mechanical component design based on genetic adaptive search, Transactions of the ASME: Journal of Mechanical Design, 120(2), pp.162-164, (1998).

[DEB99] K. Deb, Multi-Objective Genetic Algorithms : Problem Difficulties and Construction of Test Problems, Evolutionary Computation, vol. 7 (3), pp. 205-230 (1998).

[DEB99a] K. Deb, Multi-Objective Evolutionary Algorithms : Introducing Bias Among Pareto-Optimal Solutions, KanGAL report 99002, Indian Institute of Technology, Kanpur, India (1999).

[DEB99b] K. Deb, Evolutionary Algorithms for MultiCriterion Optimization in Engineering Design, In Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN'99) (1999).

[DEB00] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering, 186, pp. 311-338 (2000).

[DEB01] K. Deb, Constrained Test Problems for Multi-Objective Evolutionary Optimization, First International Conference on Evolutionary Multi-Criterion Optimization, pp. 284-298 (2001).

[DEB01a] K. Deb & S. Gulati, Design of truss-structures for minimum weight using genetic algorithms, Finite Elements in Analysis and Design 37, pp. 447-465 (2001).

[DEB01b] K. Deb, Nonlinear goal programming using multi-objective genetic algorithms, Journal of the Operational Research Society, Vol. 52 (3), pp. 291-302 (2001).

[DEB02] K. Deb, L. Thiele, M. Laumanns & E. Zitzler, Scalable Multi-Objective Optimization Test Problems, Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002), May (2002).

[DEB02a] K. Deb, A. Pratap, S. Agarwal & T. Meyarivan, A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II, IEEE Transactions on Evolutionary Computation, vol. 6(2), pp. 182-197 (2002).

[DEL01] P. De Lit, P. Latinne, B. Rekiek & A. Delchambre, Assembly planning with an ordering genetic algorithm, International Journal of Production Research, vol. 39 (16), pp. 3623-3640 (2001).

[DEL87] J.P. Delahaye, Systèmes Experts : Organisation et Programmation des Bases de Connaissance en Calcul Propositionnel, Ed. Eyrolles, Paris (1987).

[DEL88] J.P. Delahaye, Outils logiques pour l'intelligence artificielle, Ed. Eyrolles, Paris (1988).

[DOR99] M. Dorigo & G. Di Caro, The Ant Colony Optimization Meta-Heuristic, In D. Corne, M. Dorigo, and F. Glover (eds.), New Ideas in Optimization, McGraw-Hill, pp. 11-32 (1999).

[DUL02] G.S. Dulikravich, B.H. Dennis, T.J. Martin & I.N. Egorov, Multi-Disciplinary Hybrid and Evolutionary Optimization, WCCM V – Fifth World Congress on Computational Mechanics, July 7-12 (2002).

[DUY96] P. Duysinx, Optimisation topologique : du milieu continu à la structure élastique, PhD Thesis, Université de Liège (1996).

[EHR97] M. Ehrgott, Multiple Criteria Optimization : Classification and Methodology, Shaker Verlag (Berichte aus der Mathematik), Aachen (1997).

[ERI01] M. Erickson, A. Mayer & J. Horn, The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems, In E. Zitlzler, K. Deb, L. Thiele, C.A.C. Coello and D. Corne (eds.), First International Conference on Evolutionary Multicriterion Optimization, pp. 681-695, Springer-Verlag, Lecture Notes on Computer Science No. 1993 (2001).

[EZZ00] M. Ezzegaf & B. Mareschal, Utilisation d'échelles qualitatives dans les méthodes PROMETHEE, Université Libre de Bruxelles, Institut de Statistique et de Recherche Opérationnelle (2000).

[FAS99] D. Fasulo, An Analysis of Recent Work in Clustering Algorithms, University of Washington, Seattle, Department of Computer Science and Engineering, Technical Report 01-03-02, 23 pp. (1999).

[FIL02a] R. Filomeno Coelho, Ph. Bouillard & H. Bersini, Multiobjective Optimization of a Purge Valve Using a Genetic Algorithm, WCCM V – Fifth World Congress on Computational Mechanics, 7th – 12th July 2002, Vienna University of Technology, Austria, Proceedings, ISBN 3-9501554-0-6 (2002).

[FIL02b] R. Filomeno Coelho, Ph. Bouillard & H. Bersini, PAMUC – A New Method to Handle with Constraints and Multiobjectivity in Evolutionary Algorithms, IUTAM Symposium on Evolutionary Methods in Mechanics, 24th–27th September 2002, Cracow University of Technology, Cracow, Poland (2002).

[FIL03] R. Filomeno Coelho, H. Bersini & Ph. Bouillard, Parametrical Mechanical Design with Constraints and Preferences : Application to a Purge Valve, Computer Methods in Applied Mechanics and Engineering, 192/39-40, pp. 4355-4378 (2003).

[FIT90] M. Fitting, First-Order Logic and Automated Theorem Proving, Texts and Monographs in Computer Science, Springer-Verlag, New York (1990).

[FLE78] Cl. Fleury, Le dimensionnement automatique des structures élastiques, PhD Thesis, Université de Liège (1978).

[FLE00] Cl. Fleury & W.H. Zhang, Selection of appropriate approximation schemes in multi-disciplinary engineering optimization, Advances in Engineering Software 31, pp. 385-389 (2000).

[FON93] C.M. Fonseca & P.J. Fleming, Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, In S. Forrest editor, Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 416-423, San Mateo, California, University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers (1993).

[FON95] C.S Fonseca & P.J. Fleming, Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I : A Unified Formulation, University of Sheffield, UK, Research Report 564, 37 pp. (1995).

[FON95a] C.S Fonseca & P.J. Fleming, Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms II : Application Example, University of Sheffield, UK, Research Report 565, 24 pp. (1995).

[FON95b] C.S Fonseca & P.J. Fleming, An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation, vol. 3 (1), pp. 1-16 (1995).

[FRA01] Fr. Franzè & N. Speciale, A tabu-search-based algorithm for continuous multiminima problems, International Journal for Numerical Methods in Engineering, vol. 50, pp. 665-680 (2001).

[GOL89] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman, New York, 412 pp. (1989).

[GOU99] J. Goupy, Plans d'expériences pour surfaces de réponse, Editions Dunod (1999).

[GRE03] H.J. Greenberg, Mathematical Programming Glossary, World Wide Web, http://www.cu-denver.edu/~hgreenbe/glossary/ (1996-2003).

[GRO99] A.A. Groenwold, N. Stander & J.A. Snyman, A regional genetic algorithm for the discrete optimal design of truss structures, International Journal for Numerical Methods in Engineering, vol. 44, pp. 749-766 (1999).

[GUT99] W. Gutkowski, Z. Iwanow & J. Bauer, Minimum Weight Design Using Genetic Algorithm with Controlled Mutation, 3rd World Congress on Structural and Multidisciplinary Optimization (WCSMO-3), May 17-21 1999, Buffalo, New York (1999).

[HAN98] M.P. Hansen & A. Jaskiewicz, Evaluating the quality of approximations to the non-dominated set, Technical Report IMM-REP-1998-7, Technical University of Denmark, March, 31 pp. (1998).

[HAS00] O. Hasançebi & F. Erbatur, Evaluation of crossover techniques in genetic algorithm based optimum design, Computers and Structures, vol.78, pp. 435-448 (2000).

[HAY83] F. Hayes-Roth, D.A. Waterman & D.B. Lenat (eds.), Building Expert Systems, Addison-Wesley Publishing Company, Inc., Teknowledge Series in Knowledge Engineering, 444 pp. (1983).

[HIN97] R. Hinterding, Z. Michalewicz & A.E. Eiben, Adaptation in Evolutionary Computation : A Survey, Proceedings of The IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence (1997).

[HIN98] R. Hinterding & Z. Michalewicz, Yours brains and my beauty: Parent matching for constrained optimization, In Proceedings of the 5th International Conference on Evolutionary Computation, Anchorage, AK, pp. 810-815 (1998).

[HOC81] W. Hock & K. Schittkowski, Test examples for nonlinear programming codes, Springer Verlag , Berlin, 177 pp. (1981).

[HOO00] J.N. Hooker, G. Ottosson, E.S. Thornsteinsson, H.-J. Kim, A scheme for unifying optimization and constraint satisfaction methods, Knowledge Engineering Review, vol.15, pp.11-30 (2000).

[HOO02] J.N. Hooker, Logic, optimization and constraint programming, INFORMS Journal on Computing, vol.14, pp.295-321 (2002).

[HOR97] J. Horn, Multicriteria Decision Making and Evolutionary Computation, In Th. Bäck, D.B. Foge, Z. Michalewicz (eds.), The Handbook of Evolutionary Computation, Institute of Physics Publishing, Bristol, UK (1997).

[HOR97a] J. Horn, The Nature of Niching : Genetic Algorithms and the Eovolution of Optimal Cooperative Population, PhD Thesis, University of Illinois at Urbana Champaign, Urbana, Illinois, 242 pp. (1997).

[HOW93] D. Howe, Free On-Line Dictionary Of Computing (FOLDOC), Copyright 1993 by Denis Howe, World Wide Web, http://foldoc.doc.ic.ac.uk/foldoc/ (1993-2003).

[HUA97] M.-W. Huang & J.S. Arora, Optimal design with discrete variables : some numerical experiments, International Journal for Numerical Methods in Engineering, vol. 40, pp. 165-188 (1997).

[HUR01] J.E. Hurtado & D.A. Alvarez, Neural-network-based reliability analysis : a comparative study, Comput. Meth. Appl. Mech. Engrg. 191, pp. 113-132 (2001).

[IGE03] C. Igel & M. Toussaint, On Classes of Functions for which No Free Lunch Results Hold, Information Processing Letters, 86(6), pp. 317-321 (2003).

[JAS01] A. Jaszkiewicz, Multiple objective methaheuristic algorithms for combinatorial optimization, Habilitation thesis, Poznan University of Technology (2001).

[JEN97] W.M. Jenkins, On the application of natural algorithms to structural design optimization, Engineering Structures, vol. 19 (4), pp. 302-308 (1997).

[JHA95] N.K. Jha & K. Hornik, Integrated computer-aided optimal design and finite element analysis of a plain milling cutter, Appl. Math. modelling, vol. 19 (6), pp. 343-353 (1995).

[JIA00] Z. Jiang, K. Cheng & D.K. Harrison, A concurrent engineering approach to the development of a scroll compressor, Journal of Materials Processing Technology, Vol. 107 (1-3), pp. 194-200 (2000).

[JIM99] F. Jiménez & J.L. Verdegay, Evolutionary techniques for constrained optimization problems, In 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99), Aachen, Germany, Springer-Verlag (1999).

[JIN01] Y. Jin, M. Olhofer & B. Sendhoff, Evolutionary Dynamic Weighted Aggregation (EDWA) : Why Does It Work and How ?, Proceedings of Genetic and Evolutionary Computation Conference, San Francisco, USA, pp.1042-1049 (2001).

[JIN01a] Y. Jin, T. Okabe & B. Sendhoff, Adapting weighted aggregation for multi-objective evolution strategies, First International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, Springer, Zurich, pp. 96-110 (2001).

[JIN02] Y. Jin & B. Sendhoff, Incorporation of Fuzzy Preferences into Evolutionary Multiobjective Optimization, in Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'2002), vol.1, pp. 26-30, Singapore (2002).

[JOI94] J.A. Joines & C.R. Houck, On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs, Proceedings of the First IEEE International Conference on Evolutionary Computation, pp. 579-584, IEEE Press (1994).

[JON00] D. Jonson, J. DeBeer & N. Diya, Motion analysis using a logic-based modeling approach, Computational Mechanics : Techniques and Developments, Civil-Comp Press, Edinburgh, pp. 1-8 (2000).

[KAL01] L. Kallel, B. Naudts, A. Rogers, G. Rozenberg, T. Bäck, A.E. Eiben, J.N. Kok, H.P. Spaink (Eds.), Theoretical Aspects of Evolutionary Computing, Springer Verlag, 1st edition, 497 pp. (2001).

[KAM01] E.S. Kameshki & M.P. Saka, Optimum design of nonlinear steel frames with semi-rigid connections using a genetic algorithm, Computers and Structures 79, pp. 1593-1604 (2001).

[KIM99] J.-S. Kim, C.-G. Kim & C.-S. Hong, Optimum design of composite structures with ply drop using genetic algorithm and expert system shell, Composite Structures, vol. 46 (2), pp. 171-187 (1999).

[KIM02] I.Y. Kim & B.M. Kwak, Design space optimization using a numerical design continuation method, International Journal for Numerical Methods in Engineering, vol. 53, pp. 1979-2002 (2000).

[KLE67] S.C. Kleene, Mathematical Logic, John Wiley and Sons, New York, 413 pp. (1967).

[KLE96] F. J. Kleinschrodt, III & J. D. Jones, Industrial vision for process optimization, Computers & Chemical Engineering, vol. 20, Supplement 1, pp. S473-S483 (1996).

[KNO00] J.D. Knowles & D.W. Corne, Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy, Evolutionary Computation, vol. 8 (2), pp. 149-172 (2000).

[KNO02] J.D. Knowles & D.W. Corne, On Metrics for Comparing Non-Dominated Sets, Proceedings of the 2002 Congress on Evolutionary Computation Conference (CEC02), pp. 711-716. IEEE Press (2002).

[KNO02a] J.D. Knowles, Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization, PhD Thesis, Department of Computer Science, University of Reading, UK, 335 pp. (2002).

[KOU02] V.K. Koumousis & C.P. Katsaras, The Effect of Oscillating Population Size and Re-initialization on the Performance of Genetic Algorithms, Proceedings of the Third International Conference on Engineering Computational Technology, Civil-Comp Press, Scotland (2002).

[KOZ98] S. Kozieł & Z. Michalewicz, A Decoder-based Evolutionary Algorithm for Constrained Parameter Optimization Problems, Proceedings of the 5th Parallel Problem Solving from Nature, T. Bäck, A.E. Eiben, M. Schoenauer & H.-P. Schwefel (Editors), Amsterdam, September 27-30, Springer-Verlag, Lecture Notes in Computer Science, pp.231-240 (1998).

[KOZ99] S. Kozieł & Z. Michalewicz, Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization, Evolutionary Computation, vol. 7 (1), pp. 19-44 (1999).

[KUN99] S. Kundu, A Note on Optimizality vs. Stability - A Genetic Algorithm based approach, 3rd World Congress on Structural and Multidisciplinary Optimization (WCSMO-3), May 17-21 1999, Buffalo, New York (1999).

[KUR02] A. Kurpati, S. Azarm & J. Wu, Constraint handling improvements for multiobjective genetic algorithms, Struct. Multidisc. Optim. (23), Springer-Verlag, pp. 204-213 (2002).

[LAC03] D. Lacroix & Ph. Bouillard, Improved sensitivity analysis by a coupled FE-EFG method, Computers and Structures, vol. 81, pp. 2431-2439 (2003).

[LAR88] P.J.M. van Laarhoven, Theoretical and computational aspects of simulated annealing, Centrum voor Wiskunde en Informatica, Erasmus Universiteit Rotterdam, 168 pp. (1988).

[LAG02] N.D. Lagaros, M. Papadrakakis & G. Kokossalakis, Structural optimization using evolutionary algorithms, Computers and Structures, vol.80, pp. 571-589 (2002).

[LAU99] T.L. Lau, Guided Genetic Algorithm, PhD Thesis, Department of Computer Science, University of Essex, Colchester, United Kingdom (1999).

[LEE96] D. K. Lee & S.-Y. Kim, A Knowledge-Based Expert-System as a Pre-Post Processor in Engineering Optimization, Systems with Applications, vol. 11, N° 1, pp. 79-87 (1996).

[LEI98] J.P.B. Leite & B.H.V. Topping, Improved genetic operators for structural engineering optimization, Advances in Engineering Software, vol. 29 (7-9), pp. 529-562 (1998).

[LEY02] J.C. Leyva-López & E. Fernández-González, A new method for group decision support based on ELECTRE III methodology, European Journal of Operational Research, vol. 148 (1), pp. 14-27 (2003).

[LLO84] J.W. Lloyd, Foundations of logic programming, Springer-Verlag, New York, 124 pp. (1984).

[MAC01] J.M. Machado, Y. Shiyou , S.L. Ho & N. Peihong, A common Tabu seach algorithm for the globzal optimization of engineering problems, Comput. Meth. Appl. Mech. Engrg. 190, pp. 3501-3510 (2001).

[MAC02] J. Mackerle, Topology and shape optimization of structures using FEM and BEM : a bibliography (1999-2001), Finite Elements in Analysis and Design, vol. 39, pp. 243-253 (2003).

[MAC02a] Th. McAvoy, Intelligent "control" applications in the process industries, Annual Reviews in Control, vol. 26 (1), pp. 75-86 (2002).

[MAN99] S. Manoharan & S. Shanmuganathan, A comparison of search mechanisms for structural optimization, Computers and Structures, vol.73, pp. 363-372 (1999).

[MAR89] B. Mareschal, Aide à la décision multicritère : développements théoriques et applications (le système d'évaluation industrielle BANK ADVISER et le système expert CHRONOS pour la prévision), PhD Thesis, Université Libre de Bruxelles (1989).

[MAS99] S. Massebeuf, C. Fonteix, L.N. Kiss, I. Marc, F. Pla & K. Zaras, Multicriteria Optimization and Decision Engineering of an Extrusion Process Aided by a Diploid Genetic Algorithm, Congress on Evolutionary Computation, pp. 14-21, Washington, D.C., July 1999, IEEE Service Center (1999).

[MAT00] K. Matouš, M. Lepš, J. Zeman & M. Šejnoha, Applying genetic algorithms to selected topics commonly encountered in engineering practice, Comput. Meth. Appl. Mech. Engrg. 190, pp. 1629-1650 (2000).

[MEZ02] E. Mezura-Montes & C.A.C. Coello, A Numerical Comparison of some Multiobjective-Based Techniques to Handle Constraints in Genetic Algorithms, Technical Report EVOCINV-03-2002, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México (2002).

[MIC95] Z. Michalewicz, A Survey of Constraint Handling Techniques in Evolutionary Computation Methods, Proc. of the 4th Annual Conf. on Evolutionary Programming, pp. 135-155 (1995).

[MIC95a] Z. Michalewicz, Heuristic Methods for Evolutionary Computation Techniques, Journal of Heuristics, vol.1 (2), pp.177-206 (1995).

[MIC95b] Z. Michalewicz & G. Nazhiyath, Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints, In D. B. Fogel (Ed.), Proceedings of the Second IEEE International Conference on Evolutionary Computation, pp. 647--651. IEEE Press (1995).

[MIC96] Z. Michalewicz, D. Dasgupta, R.G. Le Riche, M. Schoenauer, Evolutionary algorithms for constrained engineering problems, Computers & Industrial Engineering Journal, Vol.30, No.2, pp. 851-870 (1996).

[MIC96a] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer Verlag, 387 pp (1996).

[MIL95] B.L. Miller & D.E. Goldberg, Genetic Algorithms, Tournament Selection, and the Effects of Noise, Complex Systems, vol. 9, pp. 193-212 (1995).

[MOR99] L. Moreau-Giraud & P. Lafon, Evolution strategies for optimal design of mechanical systems, 3rd World Congress on Structural and Multidisciplinary Optimization (WCSMO-3), May 17-21 1999, Buffalo, New York (1999).

[MOR02] L. Moreau-Giraud & P. Lafon, A comparison of evolutionary algorithms for mechanical design components, Engineering Optimization (34) pp. 307–322 (2002).

[NAK01] Y. Nakanishi, Application of homology theory to topology optimization of three-dimensional structures using genetic algorithms, Comput. Meth. Appl. Mech. Engrg. 190, pp. 3849-3863 (2001).

[NAN01] P. Nanakorn & K. Meesomklin, An adaptive penalty function in genetic algorithms for structural design optimization, Computers and Structures 79, pp. 2527-2539 (2001).

[NEL65] J.A. Nelder & R. Mead, A simplex method for function minimization, Computer Journal, vol. 7, pp. 308-313 (1965).

[NET97] B.D. Netten & R.A. Vingerhoeds, EADOCS: Conceptual Design in Three Phases – An Application to Fibre Reinforced Composite Panels, Engineering Applications of Artificial Intelligence, vol. 10 (2), pp. 129-138 (1997).

[NIS03] National Institute of Standards and Technology, Dictionary of algorithms and data structures (2003), World Wide Web, http://www.nist.gov/dads/HTML/metaheuristic.html (1997-2003).

[NOT02] M. Nott, R. Filomeno Coelho & A. Remouchamps, Automated Valve Design, 4th International Conference on Launcher Technology, Space Launcher Liquid Propulsion, Liège, Belgium, 3rd – 6th December (2002).

[ODE03] J.T. Oden, T. Belytschko, I. Babuska & T.J.R. Hughes, Research directions in computational mechanics, Computer Methods in Applied Mechanics and Engineering, vol. 192 (7-8), pp. 913-922 (2003).

[OLS92] D.L. Olson & J.F. Courtney Jr., Decision Support Models and Expert Systems, Maxwell Macmillan International Editions, New York, 418 pp. (1992).

[OSY99] A. Osyczka, S. Krenich & K. Karas, Optimum Design of Robot Grippers using Genetic Algorithms, Proceedings of the Third World Congress of Structural and Multidisciplinary Optimization (WCSMO), Buffalo, New York (1999).

[OSY02] A. Osyczka, Evolutionary Algorithms for Single and Multicriteria Design Optimization, Studies in Fuzzyness and Soft Computing, Physica-Verlag, Heidelberg, 218 pp. (2002).

[OZD00] L. Özdamar & M. Demirhan, Experiments with new stochastic global optimization search techniques, Computers & Operations Research 27, pp. 841-865 (2000).

[PAP00] M. Papadrakakis, N.D. Lagaros & G. Kokossalakis, Evolutionary algorithms applied to structural optimization problems, High Performance Computing for Computational Mechanics, Saxe-Cobourg Publications, Topping & Lämmer Editors, Edinburgh, pp. 207-233 (2000).

[PAR00] P.M. Pardalos, H. Edwin Romeijn & Hoang Tuy, Recent developments and trends in global optimization, Journal of Computational and Applied Mathematics, vol. 124 pp. 209-228 (2000).

[PER01] J. Périaux, H.Q. Chen, B. Mantel, M. Sfriou & H.T. Sui, Combinig game theory and genetic algorithms with application to DDM-nozzle optimization problems, Finite Elements in Analysis and Design 37, pp. 417-429 (2001).

[PHA98] Q.T. Pham, Dynamic optimization of chemical engineering processes by an evolutionary method, Computers. Chem. Engng., vol. 22 (7-8), pp. 1089-1097 (1998).

[PIR98] P. Pirjanian, Multiple Objective Action Selection & Behavior Fusion using Voting, PhD thesis, Faculty of Technical Sciences, Aalborg University, Denmark (1998).

[PUR01] R.C. Purshouse & P.J. Fleming, The Multi-Objective Genetic Algorithm Applied to Benchmark Problems – An Analysis, Technical Report No. 796, Departament of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, August (2001).

[RAF01] M.Y. Rafiq, G. Bugmann & D.J. Easterbrook, Neural network design for engineering applications, Computers and Structures 79, pp. 1541-1552 (2001).

[RAM88] A. Ramsay, Formal Methods in Artificial Intelligence, Cambridge Tracts in Theoretical Computer Science 6, Cambridge University Press, New York (1988).

[RAM96] J.V. Ramasamy & S. Rajasekaran, Artificial neural network and genetic algorithm for the design optimization of industrial roofs – a comparison, Computers and Structures, vol.58, pp. 747-755 (1996).

[RAN01] S. Ranji Ranjithan, S. Kishan Chetan & Harish K. Dakshima, Constraint Method-Based Evolutionary Algorithm (CMEA) for Multiobjective Optimization, First International Conference on Evolutionary Multi-Criterion Optimization, Springer-Verlag, Lecture Notes in Computer Science No. 1993, pp. 299-313 (2001).

[REI96] D.J. Reid, Genetic Algorithms in Constrained Optimization, Mathl. Comput. Modelling, Vol. 23, n° 5, pp. 87-111 (1996).

[REI97] D.J. Reid, Enhanced genetic operators for the resolution of discrete constrained optimization problems, Computers Ops. Res., vol. 24 (5), pp. 399-411 (1997).

[REK00] B. Rekiek, P. De Lit, F. Pellichero, Th. L'Eglise, E. Falkenauer & A. Delchambre, Dealing With User's Preferences in Hybrid Assembly Lines Design, Proceedings of the Management and Control of Production and Logistics Conference (2000).

[REK01] B. Rekiek, Assembly Line Design : Multiple Objective Grouping Genetic Algorithm and the Balancing of Mixed-Model Hybrid Assembly Line, PhD thesis, Service de Mécanique Analytique et CFAO, Université Libre de Bruxelles (2001).

[REM99] A. Remouchamps & Y. Radovcic, BOSS-Quattro : theoritical aspects about optimization methods and algorithms, Samtech S.A. 8, rue des Chasseurs Ardennais, 4031 Angleur, (1999).

[REY99] D. Reynolds, J. McConnachie, P. Bettess, W.C. Christie & J.W. Bull, Reverse adaptivity – A new evolutionary tool for structural optimization, International Journal for Numerical Methods in Engineering, vol. 45 (5), pp. 529-552 (1999).

[RIC89] T. Richardson, M.R. Palmer, G. Liepins & M. Hilliard, Some Guidelines for Genetic Algorithms with Penalty Functions, In J.D. Schaffer (ed.), Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, Morgan Kaufman Publishers, pp. 191-197 (1989).

[ROY93] B. Roy & D. Bouyssou, Aide Multicritère à la Décision : Méthodes et Cas, Collection Gestion, Série : Production et Techniques quantitatives appliquées à la gestion, Economica ed., Paris, 695 pp. (1993).

[RUD01a] G. Rudolph, Evolutionary Search under Partially Ordered Fitness Sets , Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001), ICSC Academic Press : Millet/Sliedrecht, M.F. Sebaaly (ed.), pp. 818-822 (2001).

[RUN00] Th.P. Runarsson & X. Yao, Stochastic Ranking for Constrained Evolutionary Optimization, IEEE Transactions on Evolutionary Computation, vol. 4 (3), pp. 284-294 (2000).

[SAD99] S.J. Sadjadi & K. Ponnambalam, Advances in trust region algorithms for constrained optimization, Applied Numerical Mathematics 29, pp. 423-443 (1999).

[SAD02] A.W. Sadek & M.K. Swailem, A Knowledge Based Expert System for Seismic Assessment of Existing Reinforced Concrete Buildings, Proceedings of the Third International Conference on Engineering Computational Technology, Civil-Comp Press, Scotland (2002).

[SAK00] M. Sakawa & K. Yauchi, Interactive decision making for multiobjective nonconvex programming problems with fuzzy numbers through coevolutionary genetic algorithms, Fuzzy Sets and Systems 114, pp. 151-165 (2000).

[SAL00] E. Salajegheh & J. Salajegheh, Optimum design of structures with discrete variables using approximation concepts, Identification, Control and Optimization of Engineering Structures, Civil-Comp Press, Edinburgh, pp. 85-93 (2000).

[SAV88] S. Savory (ed.), Artificial Intelligence and Expert Systems, Ellis Horwood Books in Computing Science, Chichester, UK, 278 pp. (1988).

[SCH93] M. Schoenauer & S. Xanthakis, Constrained GA Optimization, Proceedings of the Fifth International Conference on Genetic Algorithms (1993).

[SEN96] N. Senin, D.R. Wallace & M.J. Hakiela, Mixed continuous variable and catalog search using genetic algorithms, Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, August 18-22 (1996).

[SEP86] A. Sepúlveda & J.H. Cassis, An efficient algorithm for the optimum design of trusses with discrete variables, International Journal for Numerical Methods in Engineering, vol. 23, pp. 1111-1130 (1986).

[SHI00] C.J. Shih & Y.C. Yang, Mixed discrete and integer structural optimization using generalized Hopfield network based sequential unconstrained minimization technique with additional penalty strategy, Identification, Control and Optimization of Engineering Structures, Civil-Comp Press, Edinburgh, pp. 73-78 (2000).

[SOR01] G. Soremekun, Z. Gürdal, R.T. Haftka & L.T. Watson, Composite laminate design optimization by genetic algorithm with generalized elitist selection, Computers and Structures 79, pp. 131-143 (2001).

[SRI94] N. Srinivas & K. Deb, Multiple Objective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation, vol. 2 (3), pp. 221-248 (1994).

[SUR95] P.D. Surry, N.J. Radcliffe & I.D. Boyd, A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks : The COMOGA Method, In Terence C. Fogarty, editor, Evolutionary Computing. AISB Workshop, Selected Papers, Lecture Notes in Computer Science, pages 166-180. Springer-Verlag, Sheffield, U.K. (1995).

[TAI98] E.D. Taillard, L.-M. Gambardella, M. Gendreau & J.Y. Potvin, Adaptive Memory Programming : A Unified View of Meta-Heuristics, EURO XVI Conference Tutorial and Research Reviews booklet (semi-plenary sessions), Brussels (1998).

[TAP99] R.V. Tappeta & J.E. Renaud, Interactive Multiobjective Optimization Procedure with Local Preferences, 3rd World Congress on Structural and Multidisciplinary Optimization (WCSMO-3), May 17-21 1999, Buffalo, New York (1999).

[TAT95] D.M. Tate and A.E. Smith, A Genetic Approach to the Quadratic Assignment Problem, Computers and Operations Research, vol. 22, pp. 73-83 (1995).

[TEL88] E. R. Tello, Mastering AI Tools and Techniques, Howard W. Sams & Company, Indianapolis, USA, 543 pp. (1988).

[THA90] A. Thayse, A. Bruffaerts, P. Dupont, E. Henin & Y. Kamp, Approche logique de l'intelligence artificielle, vol.1 : De la logique classique à la programmation logique, Bordas, Paris, 386 pp. (1990).

[TRO97] K. Trojanowski & Z. Michalewicz, Evolutionary Algorithms and the Problem-Specific Knowledge, Proceedings of the 2nd National Conference on Evolutionary Computation and Global Optimisation, Rytro, Poland, Warsaw Univ. of Technology Press, pp. 281-292 (1997).

[ULU02] A.F. Ulusoy & F. Erbatur, Discrete and Continuous Structural Optimisation Using Evolution Strategies, Proceedings of the Third International Conference on Engineering Computational Technology, Civil-Comp Press, Scotland (2002).

[VAN96] A.H.C. van Kampen, C.S. Strom, & L.M.C. Buydens, Lethalization, Penalty and Repair Functions for Constraint Handling in the Genetic Algorithm Methodology, Chemometrics and Intelligent Laboratory Systems, pp. 55-68 (1996).

[VAN98] D.A. Van Veldhuizen & G.B. Lamont, Multiobjective Evolutionary Algorithm Research : A History and Analysis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB (1998).

[VAN99] D.A. Van Veldhuizen & G.B. Lamont, Multiobjective Evolutionary Algorithm Test Suites, Proceedings of the 1999 ACM Symposium on Applied Computing, San Antonio, Texas, eds. J. Carroll, H. Haddad, D. Oppenheim, B. Bryant & G.B. Lamont, pp. 351-357 (1999).

[VAN00] D.A. Van Veldhuizen & G.B. Lamont, On Measuring Multiobjective Evolutionary Algorithms Performance, Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1, pp. 204-211 (2000).

[VAN00a] D.A. Van Veldhuizen & G.B. Lamont, Multiobjective Evolutionary Algorithms : Analyzing the State-of-the-Art, Evolutionary Computation vol. 8 (2), pp. 125-147 (2000).

[VAN00b] D.A. Van Veldhuizen & G.B. Lamont, Evolutionary Computation and Convergence to a Pareto Front, Late Breaking Papers at the Genetic Programming 1998 Conference, University of Wisconsin, Madison, Wisconsin, USA (1998).

[VAN01] F. Vande Weyer, Méthodes globales d'Optimisation de Structures par Plans d'Expériences (Projet OSPEx), activity reports, Continuum Mechanics Department, Université Libre de Bruxelles (2000-2001).

[VIN89] Ph. Vincke, L'aide multicritère à la décision, Editions de l'Université de Bruxelles, 179 pp. (1989).

[VIS90] V. Visweswaran & C.A. Floudas, A global optimization algorithm GOP for certain classes of nonconvex NLPs : II. Application of theory and test problems, Computers and Chemical Engineering, vol. 14 (12), pp. 1419-1434 (1990).

[VOU99] C. Voudouris & E. Tsang, Guided local search and its application to the travelling salesman problem, European Journal of Operational Research, vol. 113 (2), pp. 469-499 (1999).

[WHI94] D. Whitley, V.S. Gordon & K. Mathias, Lamarckian Evolution, The Baldwin Effect and Function Optimization, In Y Davidor, H.-P. Schwefel & R. Männer (Eds.), Lecture Notes in Computer Science, 866, pp. 6-15, Springer-Verlag (1994).

[WHI96] D. Whitley, K. Mathias, S. Rana & J. Dzubera, Evaluating Evolutionary Algorithms, Artificial Intelligence, Journal, 85, pp.1-32 (1996).

[WIN95] G. Winter, J. Périaux, M. Galán & P. Cuesta, Genetic algorithms in engineering and computer science, John Wiley and Sons Ltd, Baffins Lane, Chichester, England, 480 pp. (1995).

[WOL95] W.T.M. Wolters & B. Mareschal, Novel types of sensitivity analysis for additive MCDM methods, European Journal of Operational Research, vol. 81, pp.281-290 (1995).

[WOL97] D.H. Wolpert & W.G. Macready, No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation, vol. 1 (1), pp. 67-82 (1997).

[WRI99] S. Wright & J. Nocedal, Numerical Optimization, Springer Series in Operation Research, Springer-Verlag New York (1999).

[XIA97] J. Xiao, Z. Michalewicz, L. Zhang & K. Trojanowski, Adaptive Evolutionary Planner/Navigator for Mobile Robots, IEEE transactions on evolutionary computation, vol. 1 (1), pp. 18-28 (1997).

[YAG96] M. Yagiura & T. Ibaraki, Metaheuristics as Robust and Simple Optimization Tools, Proc. International Conference of Evolutionary Computation (ICEC'96), pp. 541-546 (1996).

[YAN02] Y. Yang & C.K. Soh, Automated optimum design of structures using genetic programming, Computers and Structures 80, pp. 1537-1546 (2002).

[YOU01] H. Youssef, S.M. Sait & H. Adiche, Evolutionary algorithms, simulated annealing and tabu search : a comparative study, Engineering Applications of Artificial Intelligence 14, pp. 167-181 (2001).

[ZHA92] W.H. Zhang, Calcul des sensibilités et optimisation de forme par la méthode des éléments finis, PhD Thesis, Université de Liège (1992).

[ZHA94] H. Zhang & M. Stickel, Implementing Davis-Putnam's method , Technical Report, University of Iowa (1994).

[ZHA01a] W.H. Zhang, M. Domaszewski & Cl. Fleury, An improved weighting method with multibounds formulation and convex programming for multicriteria structural optimization, International Journal for Numerical Methods in Engineering, vol. 52, pp. 889-902 (2001).

[ZIT99] E. Zitzler & L. Thiele, Multiobjective Evolutionary Algorithms : A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation, vol. 3 (4), pp. 257-271 (1999).

[ZIT99a] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization : Methods and Applications, PhD Thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[ZIT00] E. Zitzler, L. Thiele & K. Deb, Comparison of Multiobjective Evolutionary Algorithms : Empirical Results, Evolutionary Computation, vol. 8 (2), pp. 173-195 (2000).

[ZIT01] E. Zitzler, M. Laumanns & L. Thiele. SPEA2 : Improving the Strength Pareto Evolutionary Algorithm, In K. Giannakoglou, D. Tsahalis, J. Périaux, P. Papailou, and T. Fogarty (eds.), EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, September 2001.

[ZIT02] E.Zitzler, M. Laumanns, L. Thiele, C.M. Fonseca & V. Grunert da Fonseca, Why Quality Assessment of Multiobjective Optimizers Is Difficult, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), July 2002, pp. 666-674 (2002).

[ZIT03] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca & V. Grunert da Fonseca, Performance Assessment of Multiobjective Optimizers : An Analysis and Review, IEEE Transactions on Evolutionary Computation, vol. 7 (2), pp. 117-131 (2003).

APPENDIX A

The terms $y_j(\mathbf{x})$ and $c_j(\mathbf{x})$ necessary to compute the objective function and the constraints for test case S-38IC (cf. § 4.4.1.5) are given below [DEB00] :

$y_1(\mathbf{x}) = x_2 + x_3 + 41.6,$ (A.1)
$c_1(\mathbf{x}) = 0.024\, x_4 - 4.62,$ (A.2)
$y_2(\mathbf{x}) = 12.5 / c_1(\mathbf{x}) + 12.0,$ (A.3)
$c_2(\mathbf{x}) = 0.0003535\, x_1^{\,2} + 0.5311\, x_1 + 0.08705\, y_2(\mathbf{x})\, x_1,$ (A.4)
$c_3(\mathbf{x}) = 0.052\, x_1 + 78.0 + 0.002377\, y_2(\mathbf{x})\, x_1,$ (A.5)
$y_3(\mathbf{x}) = c_2(\mathbf{x}) / c_3(\mathbf{x}),$ (A.6)
$y_4(\mathbf{x}) = 19.0\, y_3(\mathbf{x}),$ (A.7)
$c_4(\mathbf{x}) = 0.04782\, (x_1 - y_3(\mathbf{x})) + 0.1956\, (x_1 - y_3(\mathbf{x}))^2 / x_2 + 0.6376\, y_4(\mathbf{x}) + 1.594\, y_3(\mathbf{x}),$ (A.8)
$c_5(\mathbf{x}) = 100.0\, x_2,$ (A.9)
$c_6(\mathbf{x}) = x_1 - y_3(\mathbf{x}) - y_4(\mathbf{x}),$ (A.10)
$c_7(\mathbf{x}) = 0.95 - c_4(\mathbf{x}) / c_5(\mathbf{x}),$ (A.11)
$y_5(\mathbf{x}) = c_6(\mathbf{x})\, c_7(\mathbf{x}),$ (A.12)
$y_6(\mathbf{x}) = x_1 - y_5(\mathbf{x}) - y_4(\mathbf{x}) - y_3(\mathbf{x}),$ (A.13)
$c_8(\mathbf{x}) = 0.995\, (y_4(\mathbf{x}) + y_5(\mathbf{x})),$ (A.14)
$y_7(\mathbf{x}) = c_8(\mathbf{x}) / y_1(\mathbf{x}),$ (A.15)
$y_8(\mathbf{x}) = c_8(\mathbf{x}) / 3798.0,$ (A.16)
$c_9(\mathbf{x}) = y_7(\mathbf{x}) - 0.0663\, y_7(\mathbf{x}) / y_8(\mathbf{x}) - 0.3153,$ (A.17)
$y_9(\mathbf{x}) = 96.82 / c_9(\mathbf{x}) + 0.321\, y_1(\mathbf{x}),$ (A.18)
$y_{10}(\mathbf{x}) = 1.29\, y_5(\mathbf{x}) + 1.258\, y_4(\mathbf{x}) + 2.29\, y_3(\mathbf{x}) + 1.71\, y_6(\mathbf{x}),$ (A.19)
$y_{11}(\mathbf{x}) = 1.71\, x_1 - 0.452\, y_4(\mathbf{x}) + 0.58\, y_3(\mathbf{x}),$ (A.20)
$c_{10}(\mathbf{x}) = 12.3 / 752.3,$ (A.21)
$c_{11}(\mathbf{x}) = 1.74125\, y_2(\mathbf{x})\, x_1,$ (A.22)
$c_{12}(\mathbf{x}) = 0.995\, y_{10}(\mathbf{x}) + 1998.0,$ (A.23)
$y_{12}(\mathbf{x}) = c_{10}(\mathbf{x})\, x_1 + c_{11}(\mathbf{x}) / c_{12}(\mathbf{x}),$ (A.24)
$y_{13}(\mathbf{x}) = c_{12}(\mathbf{x}) - 1.75\, y_2(\mathbf{x}),$ (A.25)
$y_{14}(\mathbf{x}) = 3623.0 + 64.4\, x_2 + 58.4\, x_3 + 146312.0 / (y_9(\mathbf{x}) + x_5),$ (A.26)
$c_{13}(\mathbf{x}) = 0.995\, y_{10}(\mathbf{x}) + 60.8\, x_2 + 48.0\, x_4 - 0.1121\, y_{14}(\mathbf{x}) - 5095.0,$ (A.27)
$y_{15}(\mathbf{x}) = y_{13}(\mathbf{x}) / c_{13}(\mathbf{x}),$ (A.28)
$y_{16}(\mathbf{x}) = 148000.0 - 331000.0\, y_{15}(\mathbf{x}) + 40\, y_{13}(\mathbf{x}) - 61.0\, y_{15}(\mathbf{x})\, y_{13}(\mathbf{x}),$ (A.29)
$c_{14}(\mathbf{x}) = 2324.0\, y_{10}(\mathbf{x}) - 28740000.0\, y_2(\mathbf{x}),$ (A.30)
$y_{17}(\mathbf{x}) = 14130000.0 - 1328.0\, y_{10}(\mathbf{x}) - 531.0\, y_{11}(\mathbf{x}) + c_{14}(\mathbf{x}) / c_{12}(\mathbf{x}),$ (A.31)
$c_{15}(\mathbf{x}) = y_{13}(\mathbf{x}) / y_{15}(\mathbf{x}) - y_{13}(\mathbf{x}) / 0.52,$ (A.32)
$c_{16}(\mathbf{x}) = 1.104 - 0.72\, y_{15}(\mathbf{x}),$ (A.33)
$c_{17}(\mathbf{x}) = y_9(\mathbf{x}) + x_5\,.$ (A.34)

The values of $a[i]$ and $b[i]$ for i = 1,…,18 are given below :

$a[i] = \{ 0, 0, 17.505, 11.275, 214.228, 7.458, 0.961, 1.612, 0.146, 107.99, 922.693,$
$926.832, 18.766, 1072.163, 8961.448, 0.063, 71084.33, 2802713.0 \},$ (A.35)

$b[i] = \{ 0, 0, 1053.6667, 35.03, 665.585, 584.463, 265.916, 7.046, 0.222, 273.366, 1286.105,$
$1444.046, 537.141, 3247.039, 26844.086, 0.386, 140000.0, 12146108.0 \}.$ (A.36)

LIST OF FIGURES

LIST OF TABLES