Université de Liège
Faculté des Sciences Appliquées
Département d'Aéronautique, Spatial, Mécanique et Matériaux
Service de Turbomachines et Propulsion

Développement d'un algorithme génétique et application à des problèmes complexes d'optimisation

Mémoire présenté en vue de l'obtention
du Diplôme d'Étude Approfondies en Sciences Appliquées

par

Vincent KELNER
Ingénieur Civil Electro-Mecanicien

Septembre 2003

- *My dear Byerley, I see that you instinctively follow that great error that the Machine knows all. Let me cite you a case from my personal experience. The cotton industry engages experienced buyers who purchase cotton. Their procedure is to pull a tuft of cotton out of a random bale of a lot. They look at the that tuft and feel it, tease it, listen to the crackling perhaps as they do, touch it with their tongue, and through this procedure they will determine the class of cotton the bales represent. There are about a dozen such classes. As a result of their decisions, purchases are made at certain prices, blends are made in certain proportions. Now the buyers cannot yet be replaced by the Machine*
- *Why not ? Surely the data involved is not too complicated for it ?*
- *Probably not. But what data is this you refer to ? No textile chemist knows exactly what it is that the buyers test when he feels the tuft of cotton. Presumably there's the average length of the threads, their feel, the extent and nature of their slickness, the way they hang together and so on.*
  Several dozen items, subconsciously weighted, out of years of experience. But the quantitative nature of these tests is not know. So we have nothing to feed the Machine. Nor can the buyers explain their own judgment...
- *I see.*
- *There are innumerable cases like that. The Machine is only a tool after all, which can help humanity progress faster by tacking some of the burden of calculations and interpretations off his back. The task of the human brain remains what it has always been ; that of discovering new data to be analyzed, and of devising new concepts to be tested*

Extrait de “I, Robot”, Isaac Asimov, 1950.

Avant-propos

Les remerciements sont certainement la partie la plus agréable à rédiger dans ce type de manuscrit: ils s'effectuent lorsque le travail est terminé.

Cette recherche a été réalisée au sein du service Turbomachines et Propulsion de l'Université de Liège. Je dis "1000 mercis" à

- Olivier Léonard, professeur à la faculté des Sciences Appliquées et promoteur de ce mémoire: pour ton dynamise, ton enthousiasme et ta rigueur communicative mais aussi pour la confiance que tu me témoignes depuis plusieurs années ;
- mes collègues "turbomachinistes", Olivier, Pierre, Jean-François et Didier ; sans oublier les "aérodynamiciens", Ingrid, Didier, Pierre-Alexis, Jean-Marc et Philippe: pour l'ambiance amicale de (et hors) travail, pour votre bonne humeur (la plupart du temps), et pour m'avoir fait le café pendant tout ce temps.

L'application industrielle a été fournie par Techspace Aero. Je remercie donc

- Marc Nott et Albert Cornet, ingénieurs dans le département Lubrification: pour m'avoir accueilli dans leur équipe, et m'avoir ainsi montré "la dure réalité industrielle" ;
- François Tarnowski, ingénieur dans le département Modélisation: pour m'avoir fourni le code de performance.

Je suis également reconnaissant à la Région Wallonne d'avoir financé cette recherche dans le cadre d'un programme First-Europe.

Je remercie enfin les professeurs Crama, Fleury et Wehenkel qui ont accepté de s'intéresser à mes travaux.

Et puis, j'embrasse Christine, ma compagne: pour ses sourires, ses encouragements, sa patience, ses concepts (notamment culinaires) ... et pour toutes ses petites choses qui n'ont rien à voir avec l'optimisation.

Table des matières

Introduction

Dans la plupart des domaines, les ingénieurs sont fréquemment confrontés à des problèmes d'optimisation: améliorer les performances d'un moteur, diminuer le poids d'une structure, augmenter la résistance d'une pièce mécanique, minimiser un coût de production, etc.

Tout problème d'optimisation revient à définir une **fonction objectif** $f(\mathbf{x})$ (voir plusieurs), que l'on cherche à minimiser par rapport à tous les paramètres concernés. Ces paramètres, appelés **variables d'optimisation**, (dans un contexte industriel on parlera plutôt de variables de conception, ou de configuration, ou encore de design) constituent les inconnues du problème à résoudre et correspondent aux composantes du vecteur $\mathbf{x} \in \Re^{n_x}$:

$$\mathbf{x} = [x_1, \ldots, x_{n_x}]^T \tag{1}$$

La définition du problème d'optimisation est généralement complétée par la donnée des **contraintes** qui traduisent, par exemple, les limitations technologiques auxquelles sont soumises les variables de conception. On distingue ainsi des contraintes:

- de bornes:

$$\mathbf{x^l} \leq \mathbf{x} \leq \mathbf{x^u} \tag{2}$$

  avec

$$\mathbf{x^l} = \left[x_1^l, \ldots, x_{n_x}^l\right]^T \tag{3}$$

$$\mathbf{x^u} = \left[x_1^u, \ldots, x_{n_x}^u\right]^T \tag{4}$$

- d'égalité:

$$\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_{n_h}(\mathbf{x})]^T = \mathbf{0} \tag{5}$$

- d'inégalité:

$$\mathbf{g}(\mathbf{x}) = \left[g_1(\mathbf{x}), \ldots, g_{n_g}(\mathbf{x})\right]^T \leq \mathbf{0} \tag{6}$$

Mathématiquement parlant, un problème d'optimisation (mono-objectif) se présente donc sous la forme suivante:

$$\begin{array}{ll} \text{minimiser} & f(\mathbf{x}) \\ \text{avec} & \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{x^l} \leq \mathbf{x} \leq \mathbf{x^u} \end{array}$$

Signalons également que les fonctions objectifs seront par la suite systématiquement minimisées et ce, sans aucune perte de généralité.
En effet, les problèmes de maximisation et de minimisation sont strictement équivalents et le passage de l'un à l'autre peut s'effectuer à l'aide de la relation suivante:

$$\max f(\mathbf{x}) = -\min(-f(\mathbf{x})) \tag{7}$$

Les solutions $\mathbf{x} \in \Re^{n_x}$ qui satisfont simultanément les équations 2, 5 et 6 sont appelées **configurations admissibles**. Elles définissent l'**espace de conception** $Q_{ad} \subset \Re^{n_x}$ (appelé aussi espace admissible):

$$Q_{ad} = \left\{\mathbf{x} \in \Re^{n_x} \mid \mathbf{x^l} \leq \mathbf{x} \leq \mathbf{x^u}, \mathbf{h}(\mathbf{x}) = \mathbf{0}, \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\right\} \tag{8}$$

Parmi ces solutions, la configuration $\mathbf{x}^* \in Q_{ad}$ qui minimise la fonction objectif porte le nom d'**optimum global** .

De nombreuses techniques "classiques" permettent de résoudre le problème d'optimisation formulé précédemment. Parmi celles-ci, les **méthodes du gradient** [11, 37] sont basées sur le principe d'amélioration itérative suivant: partant d'un point $\mathbf{x}^0$, la recherche de l'optimum final $\mathbf{x}^*$ consiste a déterminer successivement des points intermédiaires $\mathbf{x}^1$, $\mathbf{x}^2$,..., $\mathbf{x}^*$. La détermination de ces points intermédiaires passe donc par l'évaluation d'une direction et d'un pas de descente, cela afin de diminuer la valeur de la fonction objectif au cours des itérations.
La direction du déplacement correspond à la direction opposée au gradient de la fonction objectif pour autant que celle-ci ne viole pas les contraintes. Auquel cas, elle peut être remplacée soit par un Lagrangien incluant les contraintes actives, soit par une pseudo fonction qui correspond à la fonction objectif augmentée des fonctions de pénalité. Le point $\mathbf{x}$ définissant la solution dans l'espace d'état évolue donc selon:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha_t \nabla f\left(\mathbf{x}^t\right) \tag{9}$$

Dans les méthodes d'ordre un, l'amplitude du déplacement (α_t) est soit fixée de manière arbitraire, soit calculée par interpolation mono-dimensionnelle. Les méthodes d'ordre deux, telle que la méthode de Newton, offrent par contre la possibilité de calculer cette amplitude à l'aide de la dérivée seconde de la fonction objectif (matrice Hessienne $\mathbf{H}$). Dans cas, l'équation 9 devient:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \mathbf{H}\left(\mathbf{x}^t\right)^{-1} \nabla f\left(\mathbf{x}^t\right) \tag{10}$$

Le calcul du gradient est quant à lui généralement effectué par une méthode de différences finies:

$$\frac{\partial f}{\partial x_i} = \frac{f\left(\mathbf{x_j}, x_i + \epsilon\right) - f\left(\mathbf{x_j}, x_i\right)}{\epsilon} \tag{11}$$

Une amélioration numérique de la méthode du gradient simple (équation 9), généralement utilisée dans la pratique, consiste à adopter une direction de déplacement qui dépend de la direction aux itérations précédentes. Par exemple, dans la méthode du gradient conjugué, on écrit:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha_t \left(\nabla f\left(\mathbf{x}^t\right) + \frac{|\nabla f\left(\mathbf{x}^t\right)|}{|\nabla f\left(\mathbf{x}^{t-1}\right)|} \nabla f\left(\mathbf{x}^{t-1}\right)\right) \tag{12}$$

Plutôt que de calculer explicitement l'inverse de la matrice Hessienne intervenant dans la méthode de Newton (équation 10), une autre modification fréquemment employée consiste à approximer, et remplacer, $\left(\mathbf{H}^t\right)^{-1}$ par une matrice $\mathbf{S}^t$ qui ne nécessite pas le calcul des dérivées secondes. On se ramène ainsi à une méthode d'ordre un, dite de quasi-Newton. Les diverses méthodes de type quasi-Newton (BFGS, DFP, etc) diffèrent essentiellement par la manière d'actualiser la matrice $\mathbf{S}^t$ au cours des itérations.

Les **méthodes d'approximations convexes**, telles que ConLin [12], MMA [34], GMMA [38] et GCMMA [35], font également partie des techniques "classiques" d'optimisation. Elle nécessitent à nouveau le calcul des dérivées de la fonction objectif mais le processus itératif conduisant à l'optimum est ici différent.
Ces méthodes consistent à remplacer le problème de départ (généralement implicite et non convexe) par une approximation explicite et convexe de celui-ci, qui est alors facile à résoudre. Le minimum obtenu lors de la résolution du problème approximé est alors utilisé comme point de départ pour une nouvelle approximation convexe, et le processus est ainsi répété jusque'à la convergence.
Les sous-problèmes explicites et convexes sont systématiquement obtenus par un développement en série de Taylor limité au premier ordre de la fonction objectif et des contraintes et ce, en terme de variables intermédiaires, propres à chacune des différentes méthodes.

A titre d'exemple, la méthode ConLin est basée sur l'emploi conjoint des variables directes x_i et des variables réciproques (appelées aussi inverses: $z_i = 1/x_i$). Le choix de la variable directe ou inverse se fait automatiquement selon le signe de la dérivée première de la fonction à approximer: si la dérivée par rapport au paramètre x_i est négative, on utilise la variable réciproque $1/x_i$; par contre, si cette dérivée est positive, le paramètre x_i reste inchangé. L'approximation convexe de la fonction objectif ou de la contrainte (notée indistinctement par $F(\mathbf{x})$) est ainsi donnée par l'expression:

$$F\left(\mathbf{x}\right) = \sum_{>0} p_i^{(t)} x_i + \sum_{<0} \frac{q_i^{(t)}}{x_i} + r^{(t)} \tag{13}$$

avec:

$$p_i^{(t)} = \frac{\partial F\left(\mathbf{x}^{(t)}\right)}{\partial x_i} \tag{14}$$

$$q_i^{(t)} = -\frac{\partial F\left(\mathbf{x}^{(t)}\right)}{\partial x_i} \left(x_i^{(t)}\right)^2 \tag{15}$$

$$r^{(t)} = F\left(\mathbf{x}^{(t)}\right) - \sum_{i=1}^{n} \left|\frac{\partial F\left(\mathbf{x}^{(t)}\right)}{\partial x_i}\right| x_i^{(t)} \tag{16}$$

Bien qu'offrant des performances remarquables (un optimum est généralement trouvé après quelques itérations) dans de nombreux cas, et par conséquent fréquemment employées, les méthodes "classiques" (du gradient et d'approximations convexes) souffrent néanmoins de deux inconvénients majeurs.

Tour d'abord, ces techniques ne garantissent nullement la convergence vers un optimum global. Elles constituent plutôt des méthodes d'optimisation locale qui convergent vers

l'optimum le plus proche du point de départ du processus itératif. De plus, une fois l'optimum local atteint, l'algorithme ne peut plus progresser. Pour améliorer l'efficacité de la méthode, on peut, bien entendu, l'appliquer plusieurs fois, avec différents points de départ choisis aléatoirement, et retenir comme solution finale le meilleur optimum local obtenu. Cependant, cette procédure augmente considérablement le temps de calcul, et ne garantit pas de trouver la configuration optimale.

En outre, basées sur le calcul des dérivées de la fonction objectif, ces techniques, ne peuvent pas être employées lorsque cette fonction présente des discontinuités. Par conséquent, elles sont incapables de traiter un problème d'optimisation présentant des variables discrètes.

Apparues dans les années 1980, les **métaheuristiques**, qui comprennent notamment le recuit simulé [21], les algorithmes génétiques [16, 17], le recherche Tabou [15], les colonies de fourmis [7], etc., sont des techniques générales qui, en s'affranchissant des inconvénients liés aux méthodes "classiques", sont dédiées aux problèmes "d'optimisation difficile". Elles présentent, en outre, des caractéristiques et des avantages communs.

Les métaheuristiques sont basées sur une analogie physique (recuit simulé), biologique (algorithmes génétiques), ou ethnologique (colonie de fourmis) qui leur permet de réaliser une exploration intelligente et guidée de l'espace de recherche.

De plus, ces méthodes sont d'ordre zéro, c'est-à-dire qu'elles ne requièrent que les valeurs de la fonction objectif et ne recourent pas au calcul de ses dérivées. Par conséquent, elles peuvent être appliquées à des fonctions discontinues, non dérivables, faisant intervenir des variables discrètes et/ou continues dans un espace d'état quelconque.

Enfin, ces techniques minimisent fortement le risque d'obtenir un optimum local. Les métaheuristiques *de voisinage* (comme le recuit simulé et la méthode Tabou) autorisent par exemple des dégradations temporaires de la configuration envisagée. A l'aide de ces mouvements de "remontée" de la fonction objectif, il devient alors possible de s'extraire du piège que représente un optimum local, pour repartir ensuite vers une zone plus prometteuse. Un mécanisme de contrôle des dégradations, spécifique à chaque métaheuristique, permet quant à lui d'éviter la divergence du procédé.
Les métaheuristiques *distribuées* (telles que algorithmes génétiques) disposent également de mécanismes qui leur permettent de s'extraire d'un optimum local. Ces mécanismes (tel que la mutation dans les algorithmes génétiques) qui affectent une solution, se superposent alors au processus collectif de lutte contre les optima locaux, que constitue l'évolution simultanée de toute une population de solutions candidates.

Signalons toutefois que ces métaheuristiques partagent aussi les mêmes inconvénients. Outre le fait qu'elles requièrent le réglage de divers paramètres intervenant dans la méthode, elles nécessitent généralement un nombre d'itérations très élevé. Par conséquent, elles ne peuvent être appliquées qu'à des problèmes pour lesquels le coût d'évaluation de la fonction objectif est faible.

Dans le cadre de ce travail, le **problème particulier** soumis par le partenaire industriel

(Techspace Aero) consiste à dimensionner de manière optimale un empilage de pompes de lubrification de turboréacteurs en fonction de spécifications relatives aux besoins du moteur dans plusieurs cas de vol. La complexité de ce problème d'optimisation réside dans la présence de plusieurs variables de conception continues et discrètes, de nombreuses contraintes à respecter, et de multiples objectifs à satisfaire. Par contre, l'évaluation des performances de chaque pompe est peu coûteuse en temps de calcul.

Au vu de leurs caractéristiques, les métaheuristiques semblent donc particulièrement bien adaptées à la résolution de ce problème industriel. Parmi ces métaheuristiques, les algorithmes génétiques ont bénéficié ces dernières années d'un véritable engouement. Comme en témoigne une abondante littérature, ils sont largement employés dans la communauté scientifique pour résoudre un grand nombre de problèmes complexes d'optimisation présentant des similitudes avec le cas qui nous occupe, et dans des domaines aussi divers que l'aérodynamique, la conception mécanique, l'optimisation des structures, etc.

Partant de ce constat, et en se concentrant sur le problème industriel, le sujet de ce travail a été défini comme étant le développement d'un algorithme génétique en vue de la résolution efficace de systèmes complexes, à paramètres discrets et continues, et devant satisfaire simultanément plusieurs critères de performances.

Chapitre 1

Les algorithmes génétiques

1.1 Introduction

L'évolution biologique a engendré depuis des millénaires des systèmes vivants, autonomes, aptes à résoudre des problèmes difficiles et capables de s'adapter continuellement à un environnement complexe, incertain et en constante transformation.
La grande variété des situations auxquelles la vie s'est adaptée laisse penser que le processus de l'Évolution est capable de résoudre de nombreuses classes de problèmes. En d'autres mots, il se caractérise par sa robustesse.
Les mécanismes à l'origine de l'Évolution reposent essentiellement sur la compétition qui sélectionne les individus les plus adaptés à leur milieu en leur assurant une descendance ainsi que sur une forme de coopération mise en œuvre par la reproduction sexuée.

Les possibilités espérées de ces mécanismes ont conduit, dès les années 1960, quelques chercheurs à vouloir les simuler afin de les appliquer à l'ingénierie. Trois écoles modélisant l'Évolution de façons différentes ont alors émergé indépendamment:

- les algorithmes génétiques (*Genetic Algorithms*, GAs) imaginés par Holland [17] et Goldberg [16],
- la programmation évolutionnaire (*Evolutionary Programming*, EP) introduite par Fogel [13],
- les stratégies d'évolution (*Evolution Strategies*, ESs), initiées par Rechenberg [28] et Shwefel [31].

Ces trois approches font partie de la classe des algorithmes évolutionnaires (*Evolutionary Algorithms*, EAs). Elles ne diffèrent que par l'absence ou la présence de tel ou tel opérateur, ou encore par des détails d'implémentation des opérateurs qu'elles emploient. Bien que les buts originels de ces algorithmes aient été différents, ils sont aujourd'hui employés essentiellement pour accomplir des tâches d'optimisation.

Les variantes que connaissent actuellement les algorithmes génétiques recouvrent les principales caractéristiques qui différenciaient à l'origine ces différentes approches.

1.2 L'analogie génétique: principes de base

Les caractéristiques héréditaires d'un être vivant dépendent exclusivement de son patrimoine génétique, ou **génotype**, constitué d'un ensemble de **chromosomes** formés de **gènes**. Ceux-ci codent des éléments du **phénotype** tel que, par exemple, la couleur des yeux de la mouche drosophile. Ces gènes sont eux-mêmes formés de longues séquences spécifiques de quatre nucléotides dont le rôle est similaire à celui d'un alphabet de quatre symboles. Ainsi, il existe une étape de décodage du génotype pour constituer le phénotype.

Dans le domaine des algorithmes génétiques, les phénotypes sont des solutions, plus ou moins performantes, à un problème posé. La première étape de l'analogie génétique consiste donc à associer un **individu i** à chaque vecteur **x** appartenant à l'espace d'état Q_{ad}. Cette correspondance, illustrée à la figure 1.1, est réalisée à l'aide d'un **codage** adéquat (binaire, réel, alphabétique, etc) des variables.

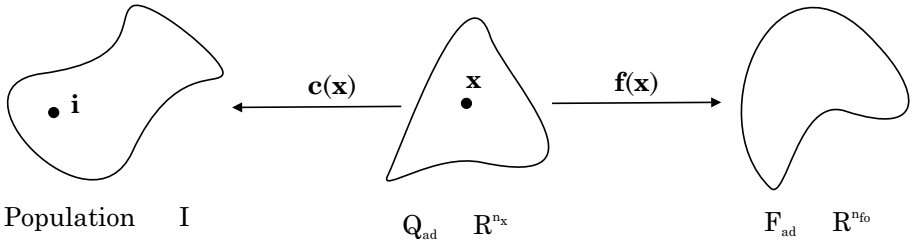


FIG. 1.1 – Relations entre population, espace de conception et espace des objectifs

Chaque solution potentielle est ainsi représentée à l'aide d'un ensemble de gènes constitués d'une chaîne de symboles appartenant à un alphabet a priori de petite taille. Ces gènes, mis bout à bout, constituent ensuite le chromosome de l'individu.
L'application de la fonction de codage $\mathbf{c}(\mathbf{x})$ à l'ensemble des solutions potentielles forme enfin l'espace des individus appelé aussi **population**.
Le tableau 1.1 résume les terminologies employées respectivement dans le cadre du processus de l'Évolution naturelle et des algorithmes génétiques.

Évolution naturelle	Algorithme génétique
gène	valeur codée d'une variable d'état
allèle	valeur réelle d'une variable d'état
chromosome	structure codée d'un point de l'espace d'état
phénotype	solution potentielle dans l'espace d'état
population	ensemble d'individus

TAB. 1.1 – Comparaison des terminologies employées

Les individus évoluent ensuite simultanément au cours des itérations sous l'action d'opérateurs génétiques qui sont inspirés de la génétique naturelle. Ils sont dénommés **opérateurs de reproduction**, opérateurs de recherche ou encore opérateurs de recombinaison. Parmi ceux-ci, on distingue:

- Le **croisement** qui consiste à obtenir deux nouveaux individus enfants en combinant les chromosomes d'une paire d'individus parents. Il s'agit là en fait de l'opérateur sous-jacent à la reproduction sexuée.
- La **mutation** qui revient à changer avec une faible probabilité un ou plusieurs symboles des gènes d'un individu.

On procède ensuite à l'**évaluation** de la qualité de chaque solution potentielle. Cette étape traduit le fait que les individus les mieux adaptés au sein d'une population sont ceux qui satisfont au mieux la fonction objectif.

Le processus de **sélection** consiste enfin à choisir, parmi tous les éléments de la population, les individus les mieux adaptés afin de les reproduire. L'opérateur de sélection est donc une version artificielle de la sélection darwinienne: dans les populations naturelles, seuls les individus les plus forts peuvent vivre jusqu'à l'âge adulte et ensuite se reproduire.

Après la création aléatoire d'une population initiale de N_{pop} individus, l'algorithme génétique accomplit une **génération** (ou itération) lorsque les opérateurs définis ci-dessus sont appliqués successivement suivant la séquence illustrée à la figure 1.2.

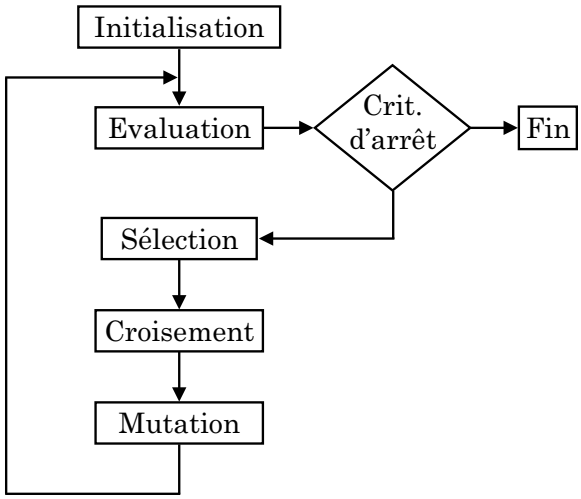


FIG. 1.2 – Schéma de fonctionnement d'un algorithme génétique

Lors d'une génération, la population est remplacée soit par la totalité de ses descendants (algorithme générationnel) ou par une partie seulement de ceux-ci (algorithme stationnaire). Néanmoins, de génération en génération, la taille de la population doit rester constante.

Différents **critères d'arrêt** peuvent être choisis. L'algorithme génétique est la plupart

du temps arrêté lorsque l'on a atteint un nombre d'itérations fixé a priori ou lorsque la population n'évolue plus assez rapidement.

Les opérateurs évoqués précédemment (codage, évaluation, sélection, croisement, mutation, arrêt) forment "l'algorithme génétique de base" (algorithme 1.1). Ils peuvent être implémentés sous de multiples formes détaillées dans les sections suivantes.

1.3 Le codage

Cette étape, réalisée directement après la phase de modélisation du problème traité, consiste à associer une structure de données aux variables d'état à l'aide d'un alphabet approprié. Bien qu'en théorie tout type d'alphabet puisse être utilisé, en pratique, seul le codage binaire et le codage réel sont généralement employés.

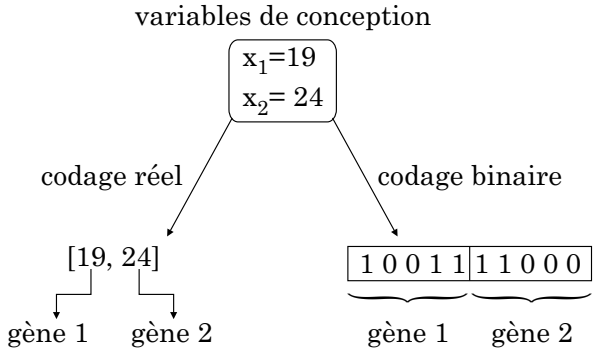


FIG. 1.3 – Exemple d'un codage réel et binaire

1.3.1 Le codage binaire

Le codage binaire fut le premier utilisé à l'origine des algorithmes évolutionnaires. C'est également en utilisant ce type de codage que les premiers résultats théoriques concernant la convergence des algorithmes génétiques ont été obtenus [16].

Comme l'illustre la figure 1.3, le codage binaire consiste à représenter chaque individu de la population au moyen d'un ensemble de digits pris dans l'ensemble $\{0, 1\}$.
Plus précisément, chaque variable d'optimisation est associée à une chaîne de l_i bits (b_{ij}) qui forment son gène. La longueur minimale de ce gène doit en outre satisfaire l'inégalité suivante:

$$2^{l_i - 1} \leq \frac{x_i^u - x_i^l}{A_c} + 1 \leq 2^{l_i} \tag{1.1}$$

où x_i^l et x_i^u désignent respectivement les bornes inférieure et supérieure de l'intervalle de définition de la variable x_i, et A_c la précision de sa représentation (c'est-à-dire une image du nombre de décimales significatives souhaitées).

Step 1	Initialisation $t = 0$ $P_t = \emptyset$ Pour $i = 1, ..., N_{pop}$ effectuer choix aléatoire de $\mathbf{i} \in I$ $P_t = P_t \cup \{\mathbf{i}\}$
Step 2	Evaluation $\forall\ \mathbf{i} \in P_t$: décodage: $\mathbf{x} = c^{-1}(\mathbf{i})$ calcul des objectifs: $\mathbf{f}(\mathbf{x})$ calcul de la fonction d'adaptation: $f_{ad}(\mathbf{i})$
Step 3	Sélection $P' = \emptyset$ $\forall\ \mathbf{i} \in P_t$: choix de $\mathbf{i} \in P_t$ selon schéma de sélection $P' = P' \cup \{\mathbf{i}\}$
Step 4	Croisement $P'' = \emptyset$ $\forall\ \mathbf{i}, \mathbf{j} \in P'$ choix aléatoire de $\mathbf{i}$ et $\mathbf{j} \in P'$ calcul de $\mathbf{k}$ et $\mathbf{l}$ à partir de $\mathbf{i}$ et $\mathbf{j}$ et selon schéma de croisement $P' = P' \setminus \{\mathbf{i}, \mathbf{j}\}$ $P'' = P'' \cup \{\mathbf{k}, \mathbf{l}\}$
Step 5	Mutation $P''' = \emptyset$ $\forall\ \mathbf{i} \in P''$ muter $\mathbf{i}$ avec une probabilité p_m et selon schéma de mutation: $\mathbf{i} \rightarrow \mathbf{i_m}$ $P''' = P''' \cup \{\mathbf{i_m}\}$
Step 6	Fin $P_{t+1} = P'''$ $t = t + 1$ Si critère d'arrêt vérifié alors stop sinon retour Step 2

ALGO. 1.1 – Pseudo code de l'algorithme génétique de base

Le chromosome associé à un point de l'espace d'état est ensuite obtenu par concaténation de ses gènes qui forment alors une chaîne de l bits:

$$l = \sum_{i=1}^{n_x} l_i \tag{1.2}$$

où n_x correspond au nombre de variables d'optimisation.

L'avantage principal du codage binaire est qu'il peut être employé quel que soit le nature de variables (réelle, entière, discrète, booléenne, chaîne de caractères, etc.). De plus, une fois l'initialisation de la population réalisée, l'emploi d'opérateurs de croisement et de mutation "classiques" permettent de traiter facilement les chaînes de bits. Néanmoins, le codage binaire souffre de nombreux inconvénients.

Une étape de décodage du chromosome de l'individu est nécessaire en vue de l'évaluation de sa fonction objectif. La conversion de la chaîne de bits relative à la variable x_i s'effectue de la façon suivante:

$$x_i = x_i^l + \left(\sum_{j=1}^{l_i} b_{ij}\ 2^{j-1} \right) \left(\frac{x_i^u - x_i^l}{2^{l_i} - 1} \right) \tag{1.3}$$

Lorsque la taille de la population et le nombre de générations nécessaires à la convergence de l'algorithme sont élevées, cette étape de décodage peut augmenter sensiblement le temps de calcul.

En outre, le processus de concaténation des gènes nécessaires à la représentation d'un individu, couplé au phénomène de discrétisation de l'espace d'état, engendre des structures codées relativement longues, ce qui rend l'étape de décodage d'autant plus coûteuse en temps de calcul.

Soit, par exemple, deux variables réelles et leur intervalle de définition respectif:

$$x_1 \in [-5, 5]$$
$$x_2 \in [-100, 100]$$

Si l'on souhaite une précision à la troisième décimale ($A_c = 0.001$) pour chacune de ces variables, la longueur respective de leur gène devra être de 14 et 18. Par conséquent, le chromosome représentatif d'une solution sera constitué d'une chaîne de 32 bits.

Le défaut de Hamming constitue un autre inconvénient inhérent au codage binaire. Il traduit une "distortion" entre l'espace de codage et l'espace d'état.

Pour comprendre ce phénomène et ses conséquences, considérons deux représentations binaires B_1 et B_2:

$$B_1\text{: } b_1 \ \ldots \ b_l \qquad B_2\text{: } b_1' \ \ldots \ b_l'$$

Définissons ensuite la distance de Hamming comme la distance entre ces deux entités binaires:

$$d_h(B_1, B_2) = \sum_{i=1}^{l} \left| b_i' - b_i \right| \tag{1.4}$$

Ainsi, à l'aide de la définition précédente et du tableau 1.2, on comprend mieux que deux points voisins dans l'espace d'état peuvent être très éloignés dans l'espace de codage (les entiers 3 et 4 ont, par exemple, une distance de Hamming égale à 3 ; alors que dans l'espace réel, ils sont distants d'une unité).
Par conséquent, les algorithmes génétiques basés sur un codage binaire sont incapables de concentrer efficacement leurs efforts de recherche dans un voisinage restreint de la population.

Le défaut de Hamming peut cependant être évité en utilisant un codage de Gray [24]. Ce dernier permet en effet, pour deux entiers consécutifs, de disposer d'une représentation binaire qui ne diffère que par la position d'un seul bit et donne par conséquent une distance de Haming toujours égale à l'unité. L'emploi d'un codage de Gray nécessite cependant une étape de codage (algorithme 1.2) et de decodage (algorithme 1.3) supplémentaire.

```
for (i = 1 ··· N_pop)
    last = 0 ;
    for (j = 1 ··· l)
        if (b_ij ≠ last)
            g_ij = 1 ;
        else
            g_ij = 0 ;
        end
        last = b_ij ;
    end
end
```

ALGO. 1.2 – Pseudo code de le conversion "binaire → Gray"

```
for (i = 1 ··· N_pop)
    last = 0 ;
    for (j = 1 ··· l)
        if (b_ij = last)
            if (last = 0)
                b_ij = 1 ;
            else
                b_ij = 0 ;
            end
        else
            b_ij = last ;
        end
        last = b_ij ;
    end
end
```

ALGO. 1.3 – Pseudo code de le conversion "Gray → binaire"

Entier	Codage binaire	Codage de Gray
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

TAB. 1.2 – Comparaison du codage binaire et du codage de Gray

1.3.2 Le codage réel

Le codage réel trouve son origine dans les techniques de programmation évolutionnaire et les stratégies d'évolution.

Comme l'illustre la figure 1.3, il consiste à représenter chaque individu de la population à l'aide d'un vecteur de nombres réels. Ainsi, il ne souffre plus des inconvénients liés au codage binaire.

En effet, la longueur du chromosome des individus est réduite au nombre de variables d'état. De plus, l'espace de représentation est quasi identique à l'espace de recherche (à l'erreur d'arrondi près). Enfin, l'étape du décodage avant l'évaluation de la fonction objectif n'est plus nécessaire.

1.4 L'opérateur de sélection

Après avoir calculé la fonction objectif de chacun des individus de la population, on procède ensuite à la sélection des meilleurs d'entre eux. Cette étape de sélection nécessite la définition préalable d'une fonction d'adaptation (f_{ad}) qui a pour but de traduire la qualité de chaque solution potentielle.
L'opérateur de sélection, indépendant du type de codage, peut être mis en œuvre sous de nombreuses formes algorithmiques détaillées dans les sections suivantes.

1.4.1 Sélection proportionnelle à l'adaptation

La sélection proportionnelle à l'adaptation, introduite par Holland [17], est la plus connue d'entre toutes. Elle consiste à attribuer à chaque individu **i** une probabilité de sélection

définie par:

$$p_s(\mathbf{i}) = \frac{f_{ad}(\mathbf{i})}{\sum_{j=1}^{N_{pop}} f_{ad}(\mathbf{j})} \tag{1.5}$$

où:

- $f_{ad}(\mathbf{i})$ désigne la fonction d'adaptation de l'individu **i**. Elle a pour but de traduire la qualité de chaque solution potentielle. De plus, telle que définie dans l'équation précédente, elle doit être nécessairement positive.
- N_{pop} est le nombre d'individus présents dans la population.

Ensuite, les parents sont choisis à l'aide d'une **roue de loterie biaisée** pour laquelle chaque individu occupe une section de roue proportionnelle à son adaptation. Chaque parent est alors sélectionné en effectuant un tour de roue.

Un exemple d'une roulette de loterie biaisée, relative à la population du tableau 1.3, est donné à la figure 1.4.

Individu	$f_{ad}(\mathbf{i})$	$p_s(\mathbf{i})$
i1	40	0.4
i2	30	0.3
i3	20	0.2
i4	10	0.1

TAB. 1.3 – Probabilité de sélection pour une population de 4 individus

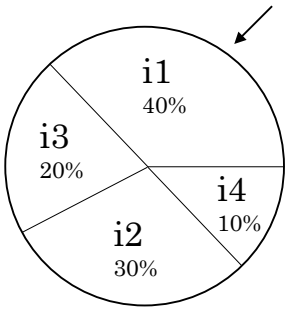


FIG. 1.4 – Une roue de loterie biaisée

L'inconvénient majeur de ce schéma de sélection est qu'il nécessite un choix de la fonction d'adaptation selon le type d'optimisation (maximisation ou minimisation) et selon les valeurs (positives ou négatives) prises par la fonction objectif (f_{obj}).
Ainsi, pour une maximisation on prend généralement:

$$f_{ad} = f_{obj} \qquad f_{obj} \geq 0 \tag{1.6}$$

$$f_{ad} = \frac{1}{\|f_{obj}\|} \qquad f_{obj} < 0 \tag{1.7}$$

alors que pour une minimisation on utilise plutôt:

$$f_{ad} = \|f_{obj}\| \qquad f_{obj} \leq 0 \tag{1.8}$$

$$f_{ad} = \frac{1}{f_{obj}} \qquad f_{obj} > 0 \tag{1.9}$$

En outre, ce processus de sélection, couplé à la roue de loterie biaisée, peut conduire à une "dérive génétique" dès les premières générations lorsque la population présente un "super élément".

Afin de minimiser ce phénomène, on préfère utiliser une **roue de loterie avec échantillonnage stochastique** qui consiste à sélectionner l'ensemble des individus à l'aide d'un seul tour de roue, celle-ci étant munie d'un nombre de pointeurs égal aux nombres d'individus souhaités lors d'un tirage.

La figure 1.5 illustre une roue de loterie avec échantillonnage stochastique destinée à sélectionner quatre individus parmi ceux présentés au tableau 1.3

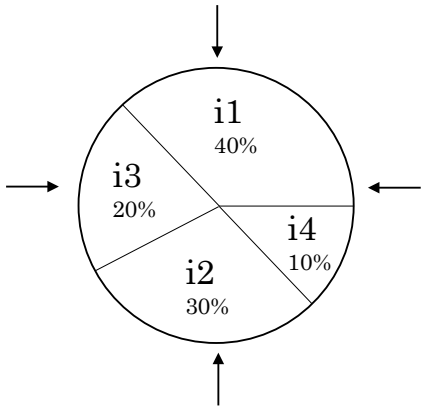


FIG. 1.5 – Une roue de loterie à 4 pointeurs offrant un échantillonnage stochastique

1.4.2 Sélection par classement

La sélection par classement consiste tout d'abord à trier les individus selon la valeur de leur fonction d'adaptation pour leur assigner ensuite un nombre R traduisant leur classement. La probabilité de sélection est alors calculée comme une fonction de ce classement. Enfin, les parents candidats à la reproduction sont sélectionnés en utilisant une roue de loterie.

Selon le type de fonction utilisée pour calculer la probabilité de sélection des individus, on distingue le classement [1]:

- linéaire:

$$p_s(\mathbf{i}) = \frac{1}{N_{pop}} \left[S_p - 2\left(S_p - 1\right) \frac{R(\mathbf{i})}{N_{pop}} \right] \tag{1.10}$$

- non-linéaire:

$$p_s(\mathbf{i}) = \frac{S_p - 1}{1 - (2 - S_p)^{N_{pop}}} (2 - S_p)^{R(\mathbf{i})-1} \tag{1.11}$$

où:

- N_{pop} est le nombre d'individus contenus dans la population candidate à la sélection ;
- $R(\mathbf{i})$ correspond au classement de l'individu $\mathbf{i}$ ($R = 1$ est attribué au meilleur élément c'est-à-dire celui satisfaisant au mieux l'objectif) ;
- S_p désigne la "pression de sélection". Ce paramètre doit être fixé dans l'intervalle $[1, 2]$. Son influence sur la probabilité de sélection des individus est illustrée à la figure 1.6.

L'avantage principal de la sélection par classement est que la fonction objectif peut toujours être assimilée à la fonction d'adaptation quel que soit le type d'optimisation et quelles que soient les valeurs prises par la fonction objectif.

Par contre, cette technique nécessite de fixer le paramètre S_p (pression de sélection). Ce dernier doit être choisi judicieusement car il influence la vitesse de convergence de l'algorithme ainsi que la qualité de l'optimum obtenu.
En effet, une sélection trop "dure" entraîne une perte de diversité de la population et un risque de convergence prématurée vers un optimum local. A l'inverse, une sélection trop "laxiste" risque d'augmenter inutilement le temps de calcul.

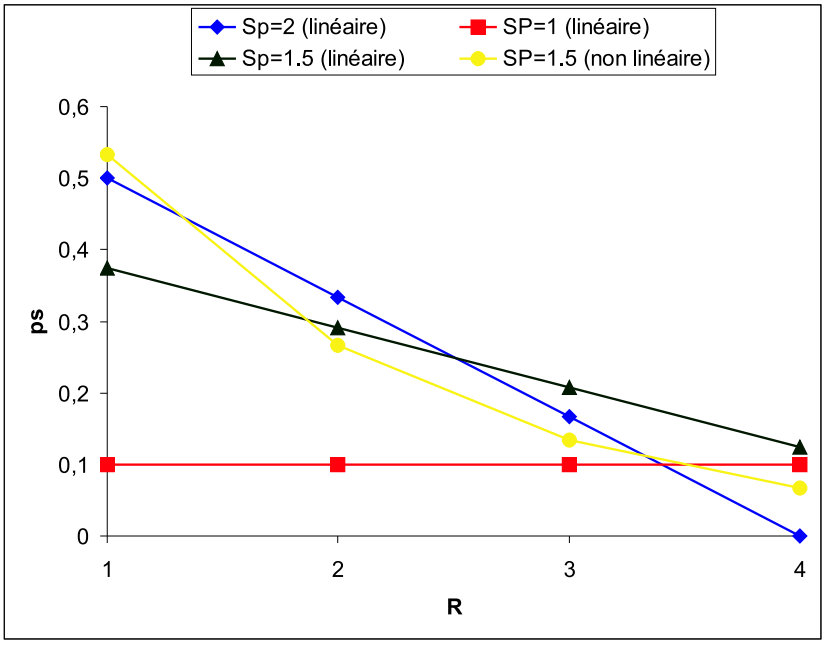


FIG. 1.6 – Probabilité de sélection fonction de la pression sélection et du classement

1.4.3 Sélection par tournoi

La sélection par tournoi consiste à sélectionner de façon aléatoire N_{tour} individus dans la population candidate à la sélection ($N_{tour} \in [2, N_{pop}]$).
Le meilleur de ces N_{tour} individus, c'est-à-dire celui possédant la meilleure fonction d'adaptation, est alors sélectionné.
Le processus est ensuite recommencé autant de fois que nécessaire (c'est-à-dire jusqu'à ce qu'une nouvelle population de N_{pop} individus soit créée), en réinjectant généralement le gagnant du tournoi dans la population.

L'avantage de ce schéma de sélection est que la fonction objectif peut à nouveau être assimilée à la fonction d'adaptation. En outre, les individus étant mis directement en compétition, il ne nécessite pas l'utilisation d'une roue de loterie.

1.5 L'opérateur de croisement

Le croisement permet la génération d'une nouvelle population d'individus (les enfants), à l'aide des chaînes les mieux adaptées (les parents) qui ont été sélectionnés lors de l'étape précédente.
La littérature fait état de nombreux schémas de croisement différents suivant le type de codage utilisé. Ils sont présentés dans les sections suivantes.

1.5.1 Croisement en un point

Le croisement en un point s'applique généralement dans le cas d'un codage binaire.
Il consiste, après avoir sélectionné deux parents, à déterminer aléatoirement un entier k compris dans l'intervalle $[1, l-1]$ où l représente la longueur du chromosome. Les enfants sont alors créés en échangeant tous les caractères compris entre les positions $k+1$ et l inclues.

Considérons par exemple les parents $\mathbf{P_1}$ et $\mathbf{P_2}$:

$$\mathbf{P_1} = 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0$$
$$\mathbf{P_2} = 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1$$

Le croisement simple ($k = 5$) illustré ci-dessous:

$$\mathbf{P_1} = 0\ 1\ 1\ 1\ 0 \,|\, 0\ 1\ 1\ 0\ 1\ 0$$
$$\mathbf{P_2} = 1\ 0\ 1\ 0\ 1 \,|\, 1\ 0\ 0\ 1\ 0\ 1$$

donne alors naissance aux deux enfants $\mathbf{E_1}$ et $\mathbf{E_2}$:

$$\mathbf{E_1} = 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1$$
$$\mathbf{E_2} = 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0$$

La figure 1.7 schématise le croisement en un point.

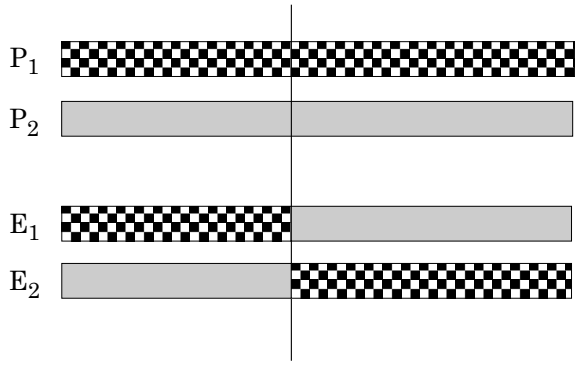


FIG. 1.7 – Principe du croisement en un point

1.5.2 Croisement multi-points

Le croisement multi-points est une variante du croisement simple et s'applique donc généralement dans le cas d'un codage binaire.
Il consiste à effectuer la permutation en plusieurs points de la chaîne de bits. Dans un premier temps, on détermine aléatoirement m entiers k_i compris dans l'intervalle $[1, l-1]$. Ces entiers correspondent aux différents points de coupure et déterminent $m+1$ portions du chromosome de chaque parent. Les enfants sont obtenus ensuite par l'échange de ces différentes portions.

Considérons à nouveau l'exemple pris précédemment:

$$\mathbf{P_1} = 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0$$
$$\mathbf{P_2} = 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1$$

Le croisement en 3 points ($k_1 = 2$, $k_2 = 6$ et $k_3 = 10$), illustré ci-dessous:

$$\mathbf{P_1} = 0\ 1\ |\ 1\ 1\ 0\ 0\ |\ 1\ 1\ 0\ 1\ |\ 0$$
$$\mathbf{P_2} = 1\ 0\ |\ 1\ 0\ 1\ 1\ |\ 0\ 0\ 1\ 0\ |\ 1$$

donne alors naissance aux deux enfants:

$$\mathbf{E_1} = 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1$$
$$\mathbf{E_2} = 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0$$

La figure 1.8 illustre schématiquement ce processus.

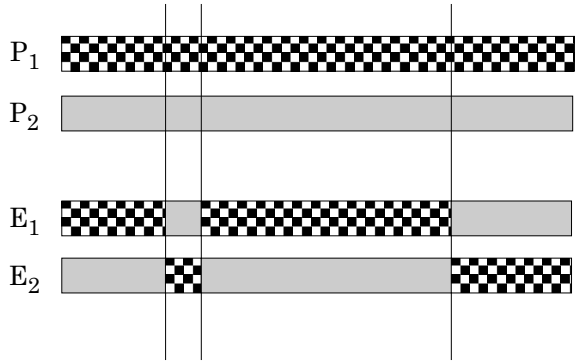


FIG. 1.8 – Principe du croisement multi-points (m=3)

1.5.3 Croisement uniforme

Le croisement uniforme généralise le croisement multi-points vu précédemment en offrant la possibilité à chacune des composantes de l'individu d'être un point de croisement éventuel. Il peut être employé dans le cas d'un codage binaire ou réel.
Ce schéma est basé sur la construction préalable d'un premier "masque de croisement", composé de bits choisis aléatoirement, et qui est de longueur identique à celle du chromosome des parents. La valeur prise par les bits de ce masque indique alors lequel des parents est candidat pour la formation du premier enfant.
Un second masque de croisement, construit par "symétrie" avec le premier, permet enfin de créer le second enfant.

Considérons à nouveau l'exemple des deux parents:

$$\mathbf{P_1} = 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0$$
$$\mathbf{P_2} = 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1$$

Déterminons aléatoirement le premier masque de croisement, ainsi que son symétrique:

$$M_1 = 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0$$
$$M_2 = 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1$$

Si l'on décide que les bits 1 et 0 dans un masque correspondent respectivement aux parents $\mathbf{P_1}$ et $\mathbf{P_2}$, les deux enfants sont alors:

$$\mathbf{E_1} = 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1$$
$$\mathbf{E_2} = 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0$$

Considérons maintenant le cas d'un codage réel où les parents sont formés de trois variables de conception:

$$\mathbf{P_1} = [-1,\ 6,\ 0.5]^T$$
$$\mathbf{P_2} = [-3,\ 4,\ 1.8]^T$$

L'application des deux masques symétriques:

$$M_1 = 0\ 0\ 1$$
$$M_2 = 1\ 1\ 0$$

permet alors de donner naissance aux deux enfants:

$$\mathbf{E_1} = [-3,\ 4,\ 0.5]^T$$
$$\mathbf{E_2} = [-1,\ 6,\ 1.8]^T$$

1.5.4 Croisement discret

Le croisement discret est une variante du croisement uniforme vu précédemment. Il peut donc être employé indistinctement dans le cas d'un codage binaire ou réel.
La différence avec le croisement uniforme réside uniquement dans la manière de fabriquer le second masque: cette fois, il n'est plus défini par symétrie avec le premier mais est généré également de façon aléatoire.

Considérons à nouveau l'exemple des parents:

$$\mathbf{P_1} = [-1,\ 6,\ 0.5]^T$$
$$\mathbf{P_2} = [-3,\ 4,\ 1.8]^T$$

Les deux masques obtenus de façon aléatoire:

$$M_1 = 0\ 0\ 1$$
$$M_2 = 0\ 1\ 1$$

permettent maintenant de donner naissance aux deux enfants:

$$\mathbf{E_1} = [-3,\ 4,\ 0.5]^T$$
$$\mathbf{E_2} = [-3,\ 6,\ 0.5]^T$$

D'un point de vue géométrique, et comme l'illustre la figure 1.9, les enfants générés par cette méthode correspondent aux coins des hypercubes défini par les parents.

1.5.5 Recombinaison intermédiaire étendue

Le croisement par recombinaison intermédiaire étendue est applicable uniquement dans le cas d'un codage réel.
Ce schéma consiste à calculer les enfants à l'aide de la relation suivante:

$$\mathbf{E_i} = \mathbf{P_1} + \mathbf{A_i}\ (\mathbf{P_2} - \mathbf{P_1}) \tag{1.12}$$

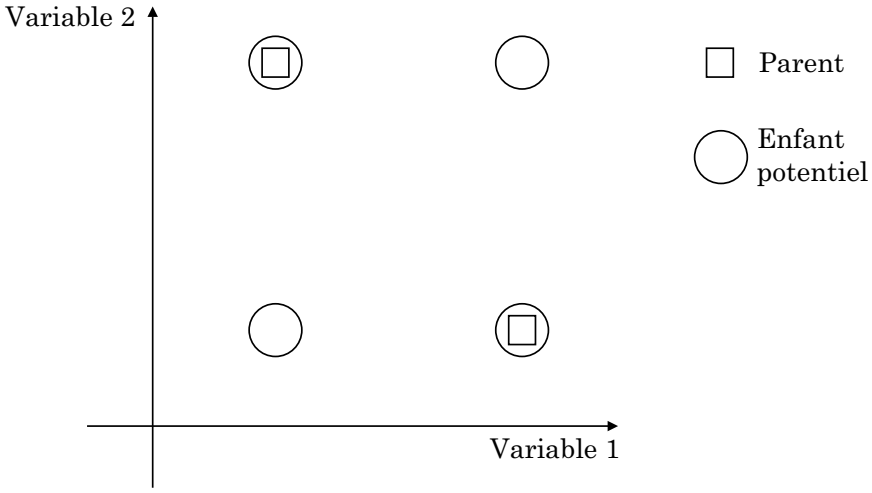


FIG. 1.9 – Interprétation géométrique du croisement discret

dans laquelle $\mathbf{A_i}$ est une matrice diagonale de dimensions $l \times l$:

$$\mathbf{A_i} = \begin{bmatrix} \alpha_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \alpha_{ll} \end{bmatrix} \tag{1.13}$$

où l désigne la longueur du chromosome, et α_{ii} sont des composantes déterminées aléatoirement dans l'intervalle $[-d, 1+d]$ avec $d \geq 0$.

Le rôle des composantes α_{ii}, par l'intermédiaire du paramètre d, est d'étendre la zone "prometteuse" d'exploitation de l'espace de conception au-delà de celle définie par les parents.

Considérons par exemple les deux parents formés de trois variables réelles:

$$\mathbf{P_1} = [12,\ 25,\ 5]^T$$
$$\mathbf{P_2} = [123,\ 4,\ 34]^T$$

Fixons ensuite $d = 0.25$, et considérons les deux matrices obtenues aléatoirement:

$$\mathbf{A_1} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1.1 & 0 \\ 0 & 0 & -0.1 \end{bmatrix} \tag{1.14}$$

$$\mathbf{A_2} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{1.15}$$

L'équation 1.12 nous permet alors de former les deux enfants:

$$\mathbf{E_1} = [67.5,\ 1.9,\ 2.1]^T$$
$$\mathbf{E_2} = [23.1,\ 8.2,\ 19.5]^T$$

D'un point de vue géométrique, et comme l'illustre la figure 1.10, le croisement par recombinaison intermédiaire étendue permet de générer des enfants dans un hypercube sensiblement plus large que celui défini par les parents.

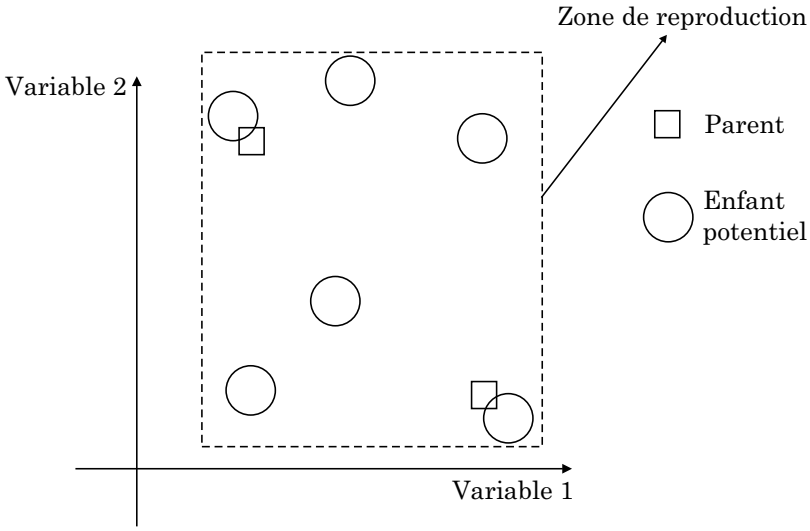


FIG. 1.10 – Interprétation géométrique de la recombinaison intermédiaire étendue

1.5.6 Recombinaison linéaire étendue

Le croisement par recombinaison linéaire étendue est une variante du précédent. Il est donc utilisable uniquement dans le cas d'un codage réel.
Ce schéma consiste à remplacer le paramètre tensoriel $\mathbf{A_i}$ par un scalaire α_i. L'équation 1.12 devient alors:

$$\mathbf{E_i} = \mathbf{P_1} + \alpha_i \ (\mathbf{P_2} - \mathbf{P_1}) \tag{1.16}$$

Considérons à nouveau l'exemple des deux parents:

$$\mathbf{P_1} = [12,\ 25,\ 5]^T$$
$$\mathbf{P_2} = [123,\ 4,\ 34]^T$$

ainsi que les paramètres scalaires obtenus aléatoirement:

$$\alpha_1 = 0.5$$
$$\alpha_2 = 0.1$$

L'équation 1.12 nous donnent ici les deux enfants:

$$\mathbf{E_1} = [67.5,\ 14.5,\ 19.5]^T$$
$$\mathbf{E_2} = [23.1,\ 22.9,\ 7.8]^T$$

D'un point de vue géométrique, et comme l'illustre la figure 1.11, ce schéma de croisement permet de générer des enfants sur la droite définie par les deux parents.

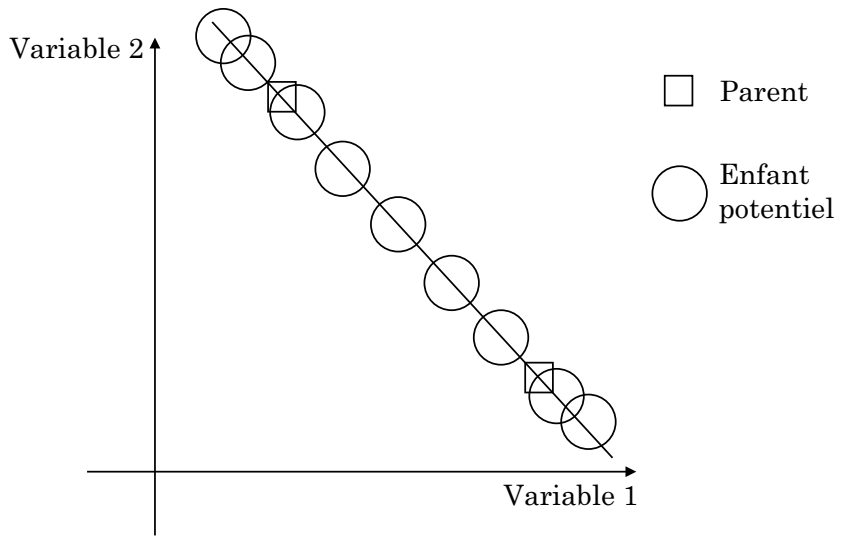


FIG. 1.11 – Interprétation géométrique de la recombinaison linéaire étendue

D'autres formulations du croisement par recombinaison linéaire étendue sont disponibles dans la littérature. Parmi celles-ci, citons celle proposée par Eshelman et Schaffer [10], plus connue sous le nom de **croisement BLX-alpha**. Ce croisement consiste à déterminer les enfants à l'aide des équations:

$$\mathbf{E_1} = \gamma\ \mathbf{P_1} + (1-\gamma)\ \mathbf{P_2} \tag{1.17}$$

$$\mathbf{E_2} = (1-\gamma)\ \mathbf{P_1} + \gamma\ \mathbf{P_2} \tag{1.18}$$

dans lesquelles le terme γ est défini par:

$$\gamma = u\ (1+2\alpha) - \alpha \tag{1.19}$$

où u est un nombre aléatoire compris dans l'intervalle $[0,1]$, et α désigne un paramètre fixé par l'utilisateur.

Comme l'illustre la figure 1.12, le rôle du paramètre α est d'étendre à nouveau la zone "prometteuse" d'exploration de l'espace de conception au-delà des celle définie par les parents.

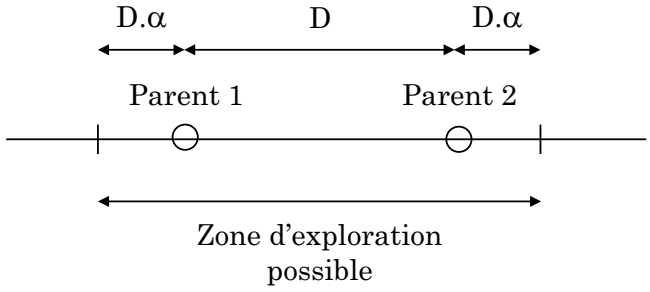


FIG. 1.12 – Le croisement BLX-alpha

1.5.7 Choix d'un opérateur de croisement

L'opérateur de croisement est généralement considéré comme l'opérateur d'exploitation. On entend par là qu'il va permettre de découvrir de meilleurs solutions en combinant les avantages des solutions déjà découvertes. Par conséquent, il doit être particulièrement efficace.

Comme nous avons pu le constater, de nombreuses variations de cet opérateur sont disponibles. Malheureusement, il n'existe aucune "recette" permettant de choisir un schéma plutôt qu'un autre.

Le choix d'un opérateur de croisement dépend en premier lieu du type de codage employé. Il est donc fortement dépendant de la nature du problème d'optimisation. Ensuite, la "règle" consiste à implémenter un opérateur de croisement offrant une *exploration* suffisante de l'espace de conception (afin de localiser la zone de l'optimum global), tout en maintenant sa *concentration* sur les zones prometteuses de cet espace (pour ne pas ralentir inutilement la convergence de l'algorithme). Par exemple, un croisement multi-points permet de générer des enfants nettement différents les uns des autres du fait du nombre de croisements possibles. Par contre, il risque de détruire le patrimoine génétique présent dans les parents. De la même façon, pour les opérateurs de croisement basés sur un codage réel, il faudra régler de façon adéquate (c'est-a-dire en fonction du problème traité) le paramètre responsable de la définition de la zone de reproduction (paramètre d pour la recombinaison intermédiaire étendue, α pour le croisement BLX-alpha, etc.).

1.6 La mutation

L'opérateur de mutation intervient directement après le processus de croisement (figure 1.2). Il s'applique donc à la nouvelle population fraîchement créée.
Cet opérateur consiste à modifier, selon une probabilité très faible, la valeur d'une composante du chromosome de l'individu. Son rôle est de maintenir une diversité au sein de la population et de permettre ainsi une exploration des zones de l'espace de recherche qui ne seraient pas représentées par les différents individus.
Selon le codage employé, l'opérateur de mutation peut revêtir différentes implémentations algorithmiques présentées ci-dessous.

1.6.1 Mutation binaire

Dans le cas d'un codage binaire, l'opérateur de mutation consiste à modifier, avec une probabilité (p_m) très faible, la valeur d'un bit dans une chaîne de caractères.

Considérons l'exemple de l'individu **i** qui, codé à l'aide de 17 bits, représente un nombre

réel à 4 décimales dans l'intervalle $[0, 10]$. Avant mutation, cet individu est noté:

$$\mathbf{i} = 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0$$

Après mutation de sa quatrième composante, l'individu devient:

$$\mathbf{i}^* = 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0$$

Signalons enfin que l'influence de cet opérateur est fonction du type de codage binaire utilisé (tableau 1.4).

Individu	Phénotype par codage binaire	Phénotype par codage de Gray
$\mathbf{i}$	5.0537	4.2887
$\mathbf{i}^*$	4.4910	3.3346

TAB. 1.4 – Influence du type de codage binaire sur l'opérateur de mutation

1.6.2 Mutation réelle

Dans le cas d'un codage réel, l'opérateur de mutation consiste à ajouter un "pas de mutation aléatoire" (S_{mut}) aux variables constituant le chromosome de chaque individu. Ce pas de mutation est généralement petit mais sa taille optimale est très difficile à déterminer car elle est fonction de la nature du problème envisagé.

Muhlenbein [25] propose néanmoins de déterminer l'individu muté à l'aide de l'expression suivante:

$$\mathbf{i}^* = \mathbf{i} \pm \mathbf{S_{mut}} \tag{1.20}$$

dans laquelle le pas de mutation est défini par:

$$\mathbf{S_{mut}} = \sum_{i=1}^{N_{pop}} 2^{-i}\delta_i \ \frac{\mathbf{D}}{2} \tag{1.21}$$

où $\mathbf{D}$ représente l'intervalle de définition des variables de l'individu $\mathbf{i}$, et δ_i vaut 1 selon une probabilité p_m et 0 dans les autres cas.

La figure 1.13 illustre l'effet d'une mutation dans le cas d'un problème à deux variables.

1.6.3 Choix du taux de mutation

Muhlenbein [25] a montré qu'un taux de mutation (p_m) inversément proportionnel au nombre de variables d'optimisation fournissait de bons résultats pour une vaste gamme de fonctions-tests.
Bäck [2] est arrivé aux mêmes conclusions. Il conseille cependant, dans le cas d'une fonction multi-modale, de prendre un taux de mutation plus élevé au cours des premières générations et ensuite de le faire décroître au fur et à mesure des itérations.

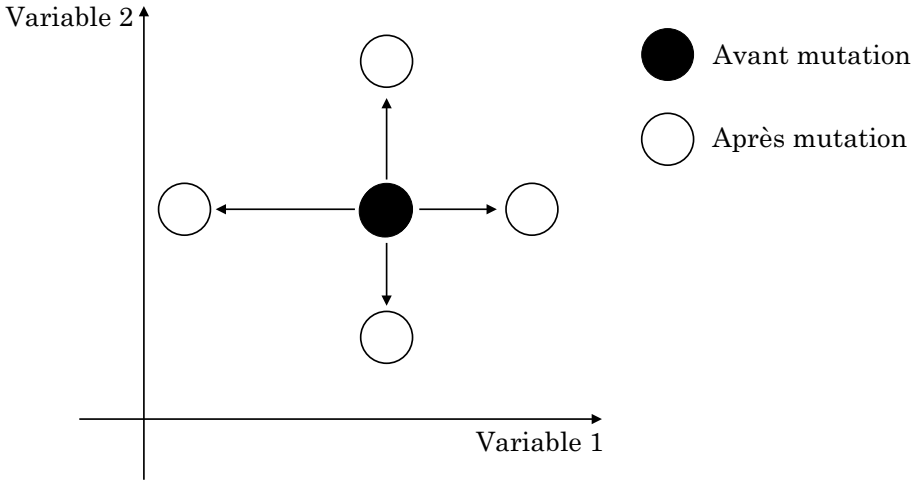


FIG. 1.13 – Interprétation géométrique de l'opérateur de mutation réelle

1.7 Le sharing

Cet opérateur a pour but de promouvoir la formation et le maintien de sous-populations stables (des niches) et ainsi, éviter une convergence prématurée de l'algorithme. Il est basé sur l'idée que les individus appartenant à une même niche doivent partager les ressources disponibles. Ainsi, plus il y a d'individus $\mathbf{j}$ dans le voisinage de l'individu $\mathbf{i}$, plus la valeur de la fonction d'adaptation de ce dernier sera dégradée. La notion de voisinage est ici définie en termes de distance $d(\mathbf{i},\mathbf{j})$ entre deux individus, et spécifiée par le paramètre σ_{sh} (rayon de niche).

Lorsque l'opérateur de sharing est appliqué, la valeur de la fonction d'adaptation de l'individu $\mathbf{i}$ devient:

$$f_{ad}^{sh}(\mathbf{i}) = m(\mathbf{i}) f_{ad}(\mathbf{i}) \tag{1.22}$$

où $m(\mathbf{i})$ désigne le compteur de niche. Celui-ci correspond à la somme des fonctions de sharing:

$$m(\mathbf{i}) = \sum_{j=1}^{N_{pop}} Sh(d(\mathbf{i},\mathbf{j})) \tag{1.23}$$

qui sont définies par:

$$Sh(d(\mathbf{i},\mathbf{j})) = \begin{cases} 0 & d(\mathbf{i},\mathbf{j}) > \sigma_{sh} \\ 1 - \frac{d(\mathbf{i},\mathbf{j})}{\sigma_{sh}} & d(\mathbf{i},\mathbf{j}) \leq \sigma_{sh} \end{cases} \tag{1.24}$$

où la distance euclidienne entre deux individus peut être calculée dans l'espace:

- des objectifs: $d(\mathbf{i},\mathbf{j}) = \|f(\mathbf{i}) - f(\mathbf{j})\|$
- de recherche: $d(\mathbf{i},\mathbf{j}) = \|x(\mathbf{i}) - x(\mathbf{j})\|$
- des individus: $d(\mathbf{i},\mathbf{j}) = \|\mathbf{i} - \mathbf{j}\|$

Signalons enfin que le sharing est un opérateur "accessoire": en effet, il n'est pas indispensable au déroulement de l'algorithme génétique. Il permet cependant d'améliorer sensiblement la qualité de la solution déterminée.

Chapitre 2

L'optimisation multi-objectifs

2.1 Introduction

L'optimisation de problèmes réels fait généralement intervenir non pas un, mais plusieurs critères qui doivent être satisfaits simultanément et qui, la plupart du temps, sont contradictoires. La solution d'un problème multi-objectifs (PMO) n'est donc plus unique mais multiple, chacune des solutions représentant en effet un compromis acceptable entre les différents objectifs contradictoires. La résolution d'un PMO s'effectue alors à l'aide d'un processus global comprenant:

- une étape de recherche au cours de laquelle un ensemble de solutions optimales est déterminé,
- une phase de décision effectuée par le concepteur qui sélectionne les solutions optimales offrant le compromis souhaité.

Selon l'ordre dans lequel ces deux étapes sont envisagées, on distingue trois méthodes de résolutions différentes [36]:

- Les techniques a priori (décision $\rightarrow$ recherche): à l'aide d'indications préalablement données par le concepteur, l'optimisation consiste ensuite à résoudre un problème mono-objectif dérivé du problème multi-objectifs initial.
  Elles seront détaillées dans la quatrième section de ce chapitre.
- Les techniques a posteriori (recherche $\rightarrow$ décision): l'optimisation multi-objectifs est réalisée jusqu'à son terme sans information préliminaire. Le résultat de la recherche est donc un ensemble de solutions optimales parmi lesquelles le concepteur effectue finalement son choix.
  Les algorithmes génétiques, de part leur principe de fonctionnement, s'apparentent à ce type de technique. Nous présenterons, dans la cinquième partie de ce chapitre,

différentes implémentations permettant de modifier l'algorithme génétique mono-objectif vu précédemment, afin qu'il prenne en compte l'aspect multi-objectifs.

- Les techniques interactives (décision $\leftrightarrow$ recherche): après chaque boucle d'optimisation, le concepteur modifie ses choix au vu des solutions candidates qui lui sont proposées.
  Ces techniques, bien qu'appliquant une approche originale, présentent l'inconvénient de monopoliser l'attention du concepteur tout au long de l'optimisation. De ce fait, elles sont rarement utilisées dans le domaine des sciences de l'ingénieur, et n'ont pas été envisagées dans le cadre de ce travail. Néanmoins, le lecteur intéressé trouvera un descriptif détaillé de ces techniques dans l'ouvrage de Eschenauer [9].

2.2 Formalisme mathématique

L'optimisation multi-objectifs (appelée aussi optimisation multi-critères ou encore multi-performances) consiste à [26]:

> déterminer un vecteur des variables de décision satisfaisant les contraintes et optimisant une fonction vectorielle dont les composantes sont les fonctions objectifs du problème. Ces dernières traduisent mathématiquement les critères de performance et sont généralement en conflit les unes par rapport aux autres. Par conséquent, le terme "optimiser" signifie plutôt "déterminer une solution rendant les valeurs de l'ensemble des fonctions objectifs acceptables pour le concepteur".

Formellement, l'optimisation d'un PMO sous contraintes consiste à:

$$\begin{array}{ll} \text{déterminer} & \mathbf{x}^* \\ \text{optimisant} & \mathbf{f}(\mathbf{x}) \\ \text{sous} & \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{x}^{\mathbf{l}} \leq \mathbf{x} \leq \mathbf{x}^{\mathbf{u}} \end{array}$$

L'application de la fonction vectorielle:

$$\mathbf{f}(\mathbf{x}) = \left[f_1(\mathbf{x}), \ldots, f_{n_{fo}}(\mathbf{x})\right]^T \tag{2.1}$$

à l'ensemble des configurations admissibles:

$$Q_{ad} = \left\{\mathbf{x} \in \Re^{n_x} \mid \mathbf{x}^{\mathbf{l}} \leq \mathbf{x} \leq \mathbf{x}^{\mathbf{u}}, \mathbf{h}(\mathbf{x}) = \mathbf{0}, \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\right\} \tag{2.2}$$

forme alors l'ensemble des objectifs réalisables:

$$F_{ad} = \mathbf{f}(Q_{ad}) \tag{2.3}$$

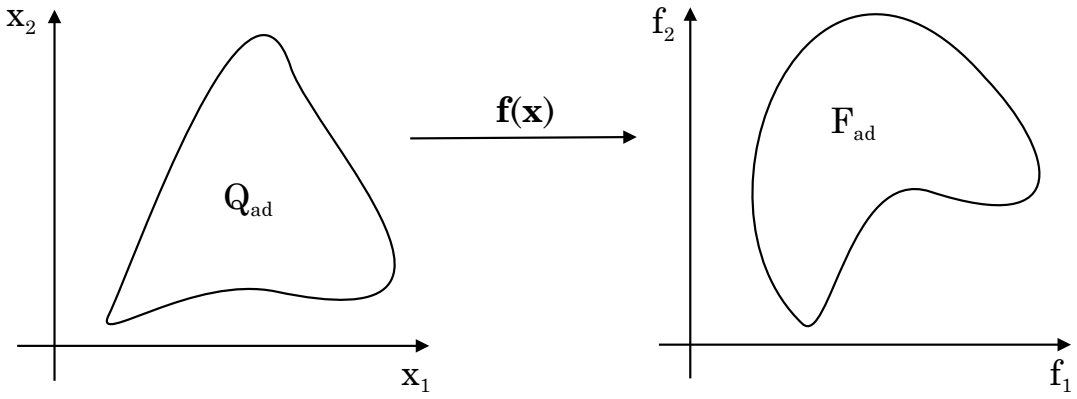


FIG. 2.1 – Q_{ad} (2 variables de conception) et F_{ad} (2 objectifs)

Ces différentes notions sont illustrées à la figure 2.1 dans le cas particulier de deux variables de conception et de deux fonctions objectifs.

Comme déjà évoqué dans l'introduction, les fonctions objectifs seront par la suite systématiquement minimisées et ce, sans aucune perte de généralité.
Ainsi, on définit une configuration $\mathbf{x^I}$ idéale comme étant celle qui permet de minimiser simultanément toutes les fonctions objectifs:

$$\forall\ k \in \{1, \ldots, n_{fo}\} : f_k(\mathbf{x^I}) = \min f_k(\mathbf{x}) \tag{2.4}$$

L'objectif idéal $\mathbf{f^I}$ qui lui correspond vaut alors:

$$\begin{aligned} \mathbf{f^I} &= \mathbf{f}(\mathbf{x^I}) \\ &= \min \mathbf{f}(\mathbf{x}) \\ &= \left[\min f_1(\mathbf{x}), \ldots, \min f_{n_{fo}}(\mathbf{x})\right]^T \end{aligned} \tag{2.5}$$

Lorsque la configuration idéale est admissible ($\mathbf{x^I} \in Q_{ad}$), elle apparaît incontestablement comme étant l'unique configuration optimale recherchée (figure 2.2).

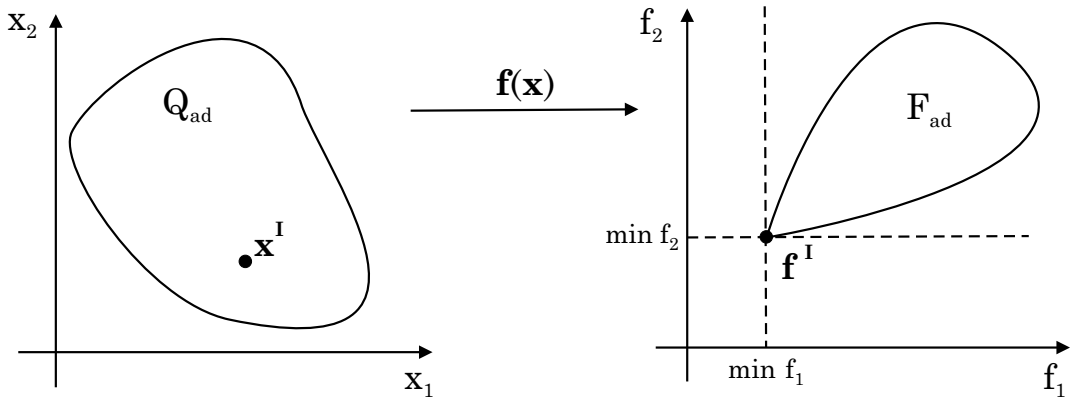


FIG. 2.2 – Exemple d'une configuration idéale appartenant à l'ensemble admissible

Cette situation est cependant rarement rencontrée dans le cas de PMO réels où la configuration idéale se trouve généralement en dehors de l'espace de conception (figure 2.3). Il convient alors d'établir un autre critère pour déterminer ce qui constitue une solution optimale.

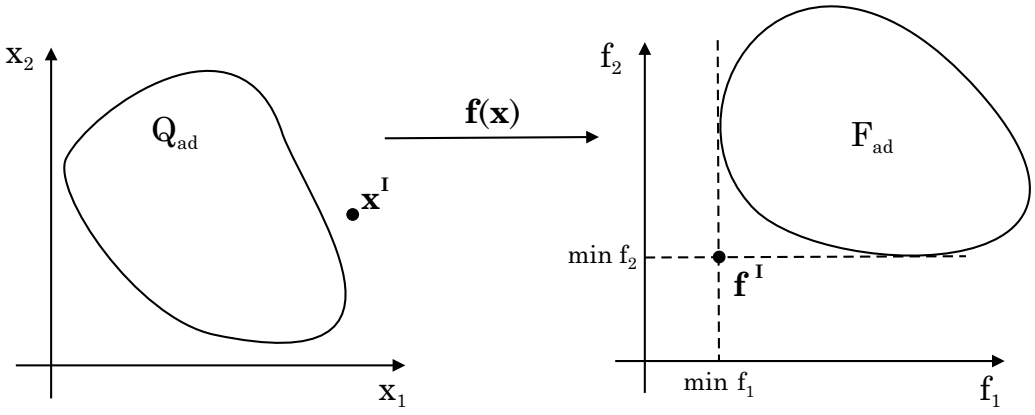


FIG. 2.3 – Exemple d'une configuration idéale située en dehors de l'ensemble admissible

2.3 Optimalité de Pareto

Le concept d'optimalité de Pareto [27] a été formulé à la fin du XIX siècle et constitue l'origine de la recherche dans le domaine de l'optimisation multi-objectifs.

Bien que fréquemment employée dans la littérature, la terminologie relative à ce concept est parfois incorrecte et souvent utilisée à mauvais escient. Afin d'assurer une consistance dans nos notations, nous rappelons ici les notions fondamentales de **dominance** et d'**optimum de Pareto** [36].

Définition 1 (Dominance de Pareto) *Le vecteur* $\boldsymbol{f^A} = \left[f_1^A, \ldots, f_{n_{fo}}^A\right]^T$ *domine le vecteur* $\boldsymbol{f^B} = \left[f_1^B, \ldots, f_{n_{fo}}^B\right]^T$ *si et seulement si* $\boldsymbol{f^A}$ *est partiellement inférieur* ($<_p$) *à* $\boldsymbol{f^B}$*:*

$$\forall\ k \in \{1, \ldots, n_{fo}\} : f_k^A \leq f_k^B \ \wedge\ \exists\ k \in \{1, \ldots, n_{fo}\} : f_k^A < f_k^B \tag{2.6}$$

Définition 2 (Optimum de Pareto) *Une solution* $\boldsymbol{x}^* \in Q_{ad}$ *est un optimum de Pareto si et seulement si il n'existe pas d'autre solution admissible* $\boldsymbol{x} \in Q_{ad}$ *pour laquelle* $\boldsymbol{f}(\boldsymbol{x})$ *domine* $\boldsymbol{f}(\boldsymbol{x}^*)$

En d'autres termes, une configuration $\mathbf{x}^*$ est optimale au sens de Pareto lorsqu'il n'existe pas d'autre configuration admissible qui améliore simultanément tous les objectifs par rapport aux valeurs obtenues pour $\mathbf{x}^*$. Pour trouver une configuration admissible offrant

de plus faibles valeurs que $\mathbf{x}^*$ pour certains objectifs, il sera donc nécessaire de dégrader au moins un des autres objectifs.

L'ensemble des solutions optimales $\mathbf{x}^*$ ainsi définies (appelées aussi parfois solutions efficaces ou non inférieures), forment l'ensemble de Pareto Q_p ($\subset Q_{ad}$):

$$Q_p = \left\{ \mathbf{x}^* \in Q_{ad} \mid \neg \exists\ \mathbf{x} \in Q_{ad} : \mathbf{f}(\mathbf{x}) <_p \mathbf{f}(\mathbf{x}^*) \right\} \tag{2.7}$$

La fonction vectorielle $\mathbf{f}(\mathbf{x})$ appliquée à l'ensemble de Pareto détermine l'ensemble des solutions non dominées appelé généralement front de Pareto F_p ($\subset F_{ad}$):

$$F_p = \mathbf{f}(Q_p) \tag{2.8}$$

$$F_p = \left\{ \mathbf{f}(\mathbf{x}^*) \in F_{ad} \mid \mathbf{f}(\mathbf{x}^*) \text{ est non dominée} \right\} \tag{2.9}$$

Ces différentes notions sont illustrées à la figure 2.4 dans le cas particulier de deux fonctions objectifs.

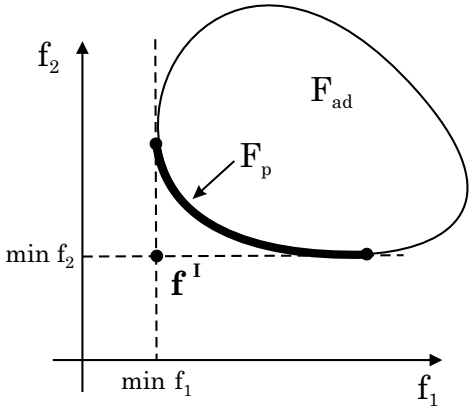
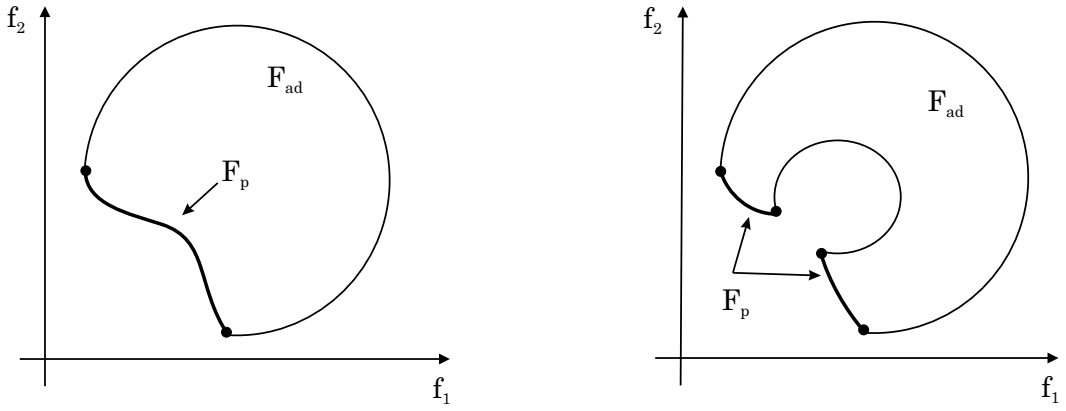


FIG. 2.4 – Exemple d'un front de Pareto



(a) F_p continu non convexe

(b) F_p discontinu

FIG. 2.5 – Irrégularités des fronts de Pareto

Remarquons enfin que l'ensemble des solutions non dominées ne possède a priori aucune propriété de régularité. En effet, un problème multi-objectifs régulier, ayant à la fois un espace de conception convexe et des fonctions objectifs continues, peut parfaitement générer un front de Pareto non convexe (figure 2.5(a)) et/ou discontinu (figure 2.5(b)).

En outre, la détermination analytique du front de Pareto est pratiquement impossible dans la majorité des cas. La résolution du problème multi-objectifs consiste alors à discrétiser F_p c'est-à dire déterminer, à l'aide d'une technique adéquate, une ou plusieurs solutions non dominées.

2.4 Les techniques a priori

Les techniques a priori sont basées sur une série de choix préliminaires effectués par le concepteur. Elles permettent ensuite de déterminer une ou plusieurs solutions non dominées via la résolution d'un problème mono-objectif dérivé du problème multi-objectifs initial. C'est essentiellement le choix de la forme prise par la fonction de préférence:

$$\Phi = \Phi\left(f_1, ..., f_{n_{fo}}\right) \tag{2.10}$$

qui différencie les méthodes présentées dans cette section.

2.4.1 Méthode des combinaisons linéaires

Dans cette méthode [4, 23], la fonction de préférence correspond à une combinaison linéaire pondérée des n_{fo} objectifs:

$$\Phi = \sum_{k=1}^{n_{fo}} w_k \, \frac{f_k(\mathbf{x})}{c_k} \tag{2.11}$$

où:

- les constantes c_k réalisent l'adimensionalisation des fonctions objectifs qui peuvent présenter des ordres de grandeur différents:

$$c_k = \max f_k \tag{2.12}$$

- les coefficients de pondération w_k permettent de privilégier certains objectifs par rapport à d'autres:

$$\sum_{k=1}^{n_{fo}} w_k = 1 \qquad \text{avec} \qquad w_k \geq 0 \tag{2.13}$$

Comme l'illustre la figure 2.6, le choix de différents vecteurs $\mathbf{w} = (w_1, ..., w_{n_{fo}})$ permet alors au concepteur de déterminer plusieurs solutions non dominées.

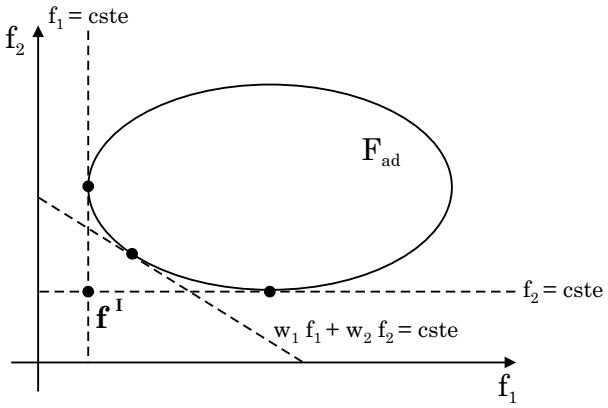


FIG. 2.6 – La méthode des combinaisons linéaires (2 fonctions objectifs)

Une convergence rapide et une implémentation très facile constituent les principaux avantages de cette méthode. Elle présente cependant plusieurs inconvénients majeurs [5, 33].

Tout d'abord, la forme du front de Pareto étant généralement inconnue a priori, un choix pertinent des valeurs des coefficients de pondération est assez difficile. Par conséquent, la répartition des points sur ce front est rarement homogène: les solutions non dominées sont regroupées dans une même zone ce qui présente peu d'intérêt pour le concepteur.

Par ailleurs, cette méthode ne permet pas d'identifier les portions non convexes du front de Pareto. En effet, comme on peut le constater sur la figure 2.7, aucun des points de l'arc compris entre $\mathbf{f^A}$ et $\mathbf{f^B}$ ne peut être déterminé par la minimisation d'une combinaison linéaire des deux fonctions objectifs.

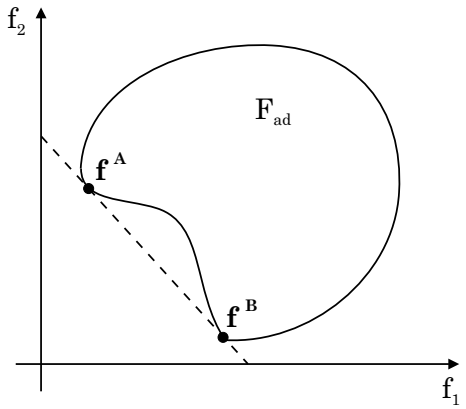


FIG. 2.7 – La méthode des combinaisons linéaires échoue si F_p est non convexe

Enfin, l'étape d'adimensionalisation nécessite l'évaluation préalable du maximum de chacune des fonctions objectifs. Dans la pratique, ce calcul peut cependant être évité en exprimant les constantes d'adimensionalisation par:

$$c_k = f_k(\mathbf{x^0}) \tag{2.14}$$

où $\mathbf{x^0}$ désigne un point quelconque de l'espace de conception.

2.4.2 Méthode des distances

Cette méthode [22, 23] consiste à déterminer la configuration admissible la plus proche possible de l'objectif idéal.

Comme l'indique la figure 2.8, la proximité entre une solution candidate $\mathbf{f}(\mathbf{x})$ et l'objectif idéal $\mathbf{f^I}$ se détermine à l'aide d'une des distances usuelles de $\Re^{n_{fo}}$:

$$d(\mathbf{x}) = \|\mathbf{f}(\mathbf{x}) - \mathbf{f}^I\|_p = \left\{ \sum_{k=1}^{n_{fo}} |f_k(\mathbf{x}) - \min f_k|^p \right\}^{\frac{1}{p}} \tag{2.15}$$

avec:

$$1 \leq p < +\infty \tag{2.16}$$

Dans la méthode des distances, la fonction de préférence s'écrit alors:

$$\Phi = \left\{ \sum_{k=1}^{n_{fo}} w_k \left| \frac{f_k(\mathbf{x}) - \min f_k}{c_k} \right|^p \right\}^{\frac{1}{p}} \tag{2.17}$$

où w_k et c_k désignent respectivement les coefficients de pondération et les constantes d'adimensionalisation qui valent ici:

$$c_k = \max f_k - \min f_k \tag{2.18}$$

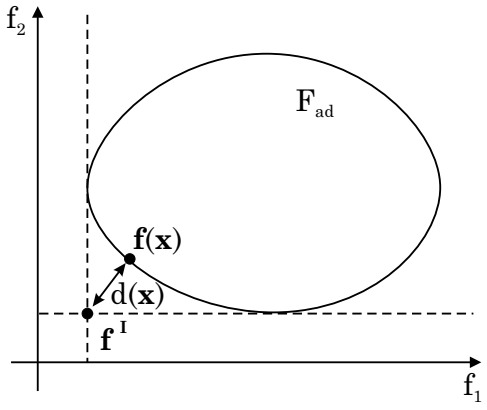


FIG. 2.8 – La méthode des distances (2 fonctions objectifs)

Signalons que dans le cas particulier de la norme infinie ($p \to +\infty$), cette méthode porte le nom spécifique de **méthode min-max**, et la fonction de préférence s'écrit alors:

$$\Phi = \max_{k=1,\ldots,n_{fo}} w_k \left| \frac{f_k(\mathbf{x}) - \min f_k}{\max f_k - \min f_k} \right| \tag{2.19}$$

Outre le fait qu'elle soit facile à implémenter, la méthode des distances présente surtout l'avantage de pouvoir identifier les portions non convexes du front de Pareto (figure 2.9).

Cette technique semble par contre moins efficace que la méthode des combinaisons lorsque les fonctions objectifs sont fortement non linéaires. De plus, elle est légèrement plus coûteuse en temps de calcul.

Par ailleurs, une répartition homogène des points sur le front de Pareto est toujours aussi difficile à obtenir si nous ne disposons pas d'indication préalable quant à sa forme.

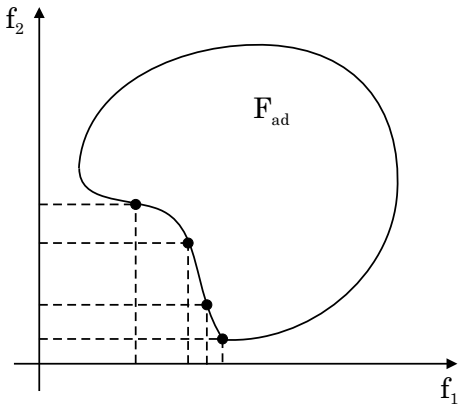


FIG. 2.9 – Discrétisation du front de Pareto par la méthode des distances

2.4.3 Méthode des contraintes

Dans cette méthode [4, 23], appelée aussi méthode du compromis, la fonction de préférence correspond à l'objectif jugé prioritaire par le concepteur:

$$\Phi = f_r(\mathbf{x}) \tag{2.20}$$

Les autres objectifs doivent quant à eux satisfaire les contraintes d'inégalité suivantes:

$$\forall\ i \in \{1, \ldots, r-1\} \cup \{r+1, \ldots, n_{fo}\} : f_i(\mathbf{x}) \leq \varepsilon_i \tag{2.21}$$

La minimisation de la fonction de préférence pour différentes valeurs de ε_i choisies par le concepteur dans l'intervalle:

$$\underline{\varepsilon}_i \leq \varepsilon_i \leq \overline{\varepsilon}_i \tag{2.22}$$

permet alors de déterminer un ensemble de solutions non dominées.

Les bornes $\underline{\varepsilon}_i$ et $\overline{\varepsilon}_i$ sont généralement définies à l'aide des relations suivantes:

$$\underline{\varepsilon}_i = \min f_i(\mathbf{x}) \tag{2.23}$$

$$\overline{\varepsilon}_i = f_i(\mathbf{x}^*_{\mathbf{r}}) \tag{2.24}$$

$$\mathbf{x}^*_{\mathbf{r}} = \min f_r(\mathbf{x}) \tag{2.25}$$

La figure 2.10 illustre la minimisation de la fonction objectif f_2 sous la restriction $f_1 \leq \varepsilon_1$. On constate également que le balayage de la contrainte ε_1 dans l'intervalle $[\underline{\varepsilon}_1, \overline{\varepsilon}_1]$ permet d'obtenir plusieurs solutions appartenant au front de Pareto.

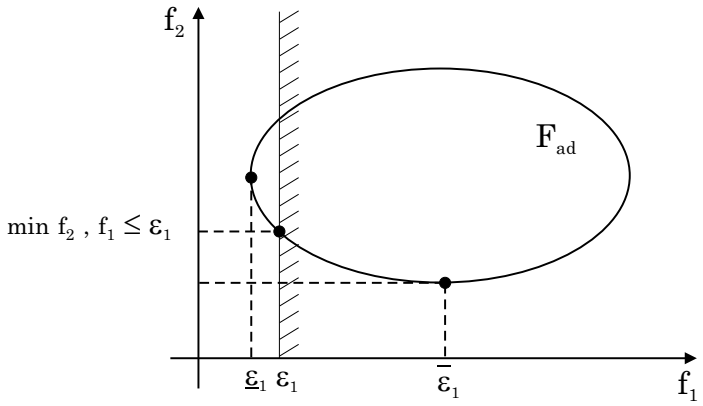


FIG. 2.10 – La méthode des contraintes (2 fonctions objectifs)

Outre le fait qu'elle soit capable de discrétiser les portions non convexes du front de Pareto, la méthode des contraintes a l'avantage d'offrir une répartition homogène des solutions non dominées *selon les directions respectives des fonctions contraintes*. Pour cela, il suffit en effet d'imposer des niveaux de contraintes ε_i uniformément réparti dans l'intervalle $[\underline{\varepsilon}_1, \overline{\varepsilon}_1]$. Néanmoins, cette approche ne garantit pas d'obtenir une répartition homogène des solutions *selon la direction de la fonction objectif prioritaire*.

2.4.4 Méthode de l'ordonnancement lexicographique

Dans cette méthode [4], le concepteur doit préalablement trier les objectifs par ordre décroissant de priorité (f_1 à $f_{n_{fo}}$). Ensuite, les fonctions sont successivement minimisées selon le classement établi.
Le problème d'optimisation consiste alors à:

$$\begin{array}{lll} \text{minimiser} & f_i(\mathbf{x}) & i = 1, \ldots, n_{fo} \\ \text{sous} & \mathbf{h}(\mathbf{x}) = \mathbf{0} & \\ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} & \\ & \mathbf{x^l} \leq \mathbf{x} \leq \mathbf{x^u} & \\ & f_l(\mathbf{x}) = f_l(\mathbf{x_l^*}) & l = 1, \ldots, i-1 \end{array}$$

Dans le formalisme précédent, remarquons que l'expression:

$$f_l(\mathbf{x}) = f_l(\mathbf{x_l^*}) \tag{2.26}$$

où:

$$\mathbf{x_l^*} = \min f_l(\mathbf{x}) \tag{2.27}$$

introduit i-1 contraintes additionnelles. Celles-ci garantissent le fait que l'optimum calculé pour f_i minimise également les i-1 fonctions objectifs précédentes.

Bien que cette méthode soit capable d'identifier les portions non convexes du front de Pareto, elle est rarement utilisée car elle présente de nombreux inconvénients.

Outre le fait qu'elle soit difficile à implémenter, cette technique nécessite un réordonancement complet des objectifs lorsque l'on souhaite obtenir une autre solution optimale. Cette étape de tri peut devenir fastidieuse lorsque le nombre de fonctions est élevé.

De plus, l'analyse combinatoire indique que le nombre maximum de solutions identifiables sur le front est limité au factoriel du nombre de fonctions objectifs. Ainsi, dans un problème présentant par exemple deux fonctions objectifs, le front de Pareto ne pourra être discrétisé que par 6 points.

2.5 Les algorithmes génétiques comme technique a posteriori

Les techniques a posteriori sont basées sur une optimisation simultanée de toutes les fonctions objectifs. A la fin du processus de recherche, le concepteur sélectionne parmi l'ensemble des solutions optimales proposées, celles lui offrant le compromis souhaité.

Les algorithmes génétiques, en mettant en œuvre un ensemble de solutions candidates qui évoluent et s'améliorent de génération en génération, constituent donc une excellente technique a posteriori. Différentes implémentations permettant la prise en compte de l'aspect multi-objectifs par ce type d'algorithmes sont présentées dans les sections suivantes.

2.5.1 VEGA

L'algorithme VEGA (*Vector Evaluated Genetic Algorithm*) a été proposé en 1985 par Schaffer [30].

Cette technique consiste à prendre en compte l'aspect multi-objectifs au travers des opérateurs d'évaluation et de sélection (algorithme 2.1) qui sont associés de la façon suivante:

- Les n_{fo} fonctions objectifs créent tour à tour une sous-population contenant $\frac{N_{pop}}{n_{fo}}$ individus obtenus à l'aide d'un schéma de sélection proportionnelle à l'adaptation.
- Par la suite, les sous-populations sont mélangées et les opérateurs de recombinaison (croisement et mutation) sont enfin appliqués à la totalité de la population.

La figure 2.11(a) illustre le mécanisme de sélection dans le cas d'une minimisation de deux fonctions objectifs. On constate que les deux-sous populations contiennent respectivement les individus qui satisfont au mieux le premier et le second objectif.

Cette procédure est facile à implémenter et ne nécessite que très peu de modifications de

l'algorithme mono-objectif existant.

Richardson [29] note cependant que dans cet algorithme, le mélange des sous-populations revient à attribuer à chaque individu une fonction d'adaptation définie par:

$$f_{ad}(\mathbf{i}) = \frac{\sum_{k=1}^{n_{fo}} f_{ad,k}(\mathbf{i})}{N_{pop}} \tag{2.28}$$

où $f_{ad,k}(i)$ désigne la fonction d'adaptation de l'individu $\mathbf{i}$ par rapport à l'objectif k. Dans la mesure où Schaffer utilise un schéma de sélection proportionnelle, la fonction d'adaptation des individus s'apparente donc à une combinaison linéaire des objectifs où les poids dépendent de la distribution de la population à chaque génération. Par conséquent, l'algorithme VEGA est inefficace lorsque le front de Pareto est non-convexe.

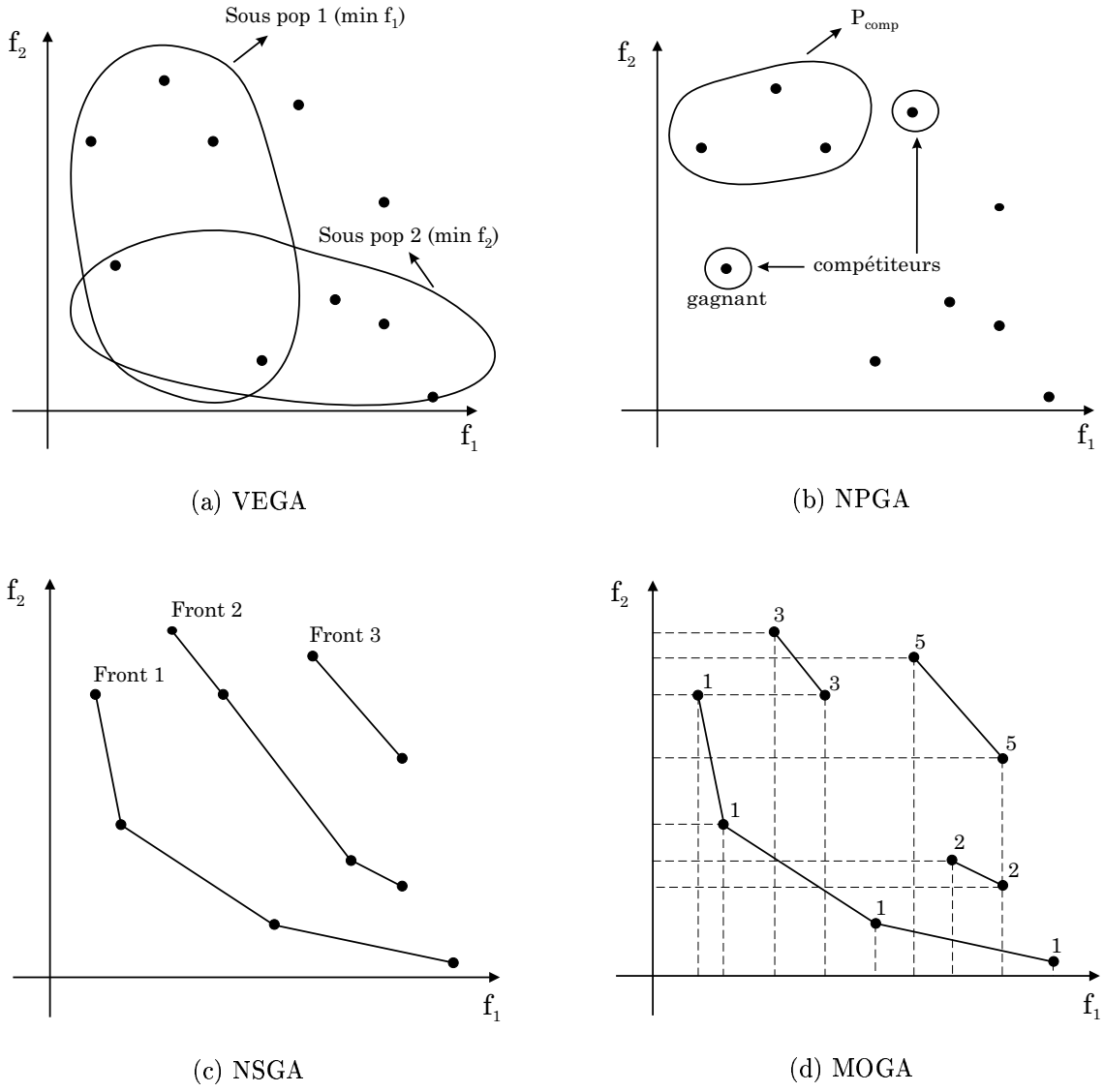


FIG. 2.11 – Différents mécanismes de sélection

Step 2.1	$k = 1$ $P' = \emptyset$
Step 2.2	$\forall$ **i** $\in P_t$: décodage: $\mathbf{x} = c^{-1}(\mathbf{i})$ calcul de la fonction objectif: $f_k(\mathbf{x})$ calcul de la fonction d'adaptation: $f_{ad}(\mathbf{i}) = \frac{1}{f_k(\mathbf{x})}$ (! minimisation)
Step 3.1	Pour $i = 1, ..., \frac{N_{pop}}{n_{fo}}$ effectuer choix de **i** $\in P_t$: sélection proportionnelle à l'adaptation $P' = P' \cup \{\mathbf{i}\}$
Step 3.2	$k = k + 1$ Si $k \leq n_{fo}$ alors retour Step 2.2 sinon mélange de P'

ALGO. 2.1 – Pseudo code de l'algorithme VEGA

2.5.2 NPGA

L'algorithme NPGA (*Niched Pareto Genetic Algorithm*) a été proposé en 1993 par Horn et Nafpliotis [18].

A l'aide de la séquence suivante, cette technique combine astucieusement l'étape de sélection par tournoi et la notion de dominance de Pareto (algorithme 2.2):

- On choisit aléatoirement deux candidats à la sélection ainsi qu'un ensemble P_{comp} formés par n_{comp} individus de la population.
- Lorsqu'un seul des deux candidats est non dominé par rapport aux éléments de P_{comp}, il est le gagnant du tournoi.
- Si les deux compétiteurs sont non dominés (ou dominés), une étape de sharing quelque peu modifiée est appliquée. Le vainqueur est alors celui qui compte dans son voisinage le plus petit nombre d'individus appartenant à la portion de population déjà sélectionnée.

La figure 2.11(b) illustre ce processus de sélection dans la cas d'une minimisation de deux fonctions objectifs.

Comme le fait remarquer Coello [4], l'algorithme NPGA est très rapide: en effet, le schéma de sélection basé sur la notion de dominance ne s'applique ici qu'à une portion seulement de la population (l'ensemble P_{dom}). Il nécessite par contre l'emploi d'un nouveau paramètre de réglage (n_{comp}) qui influence les performances de l'algorithme.

Step 2.1	$\forall\ \mathbf{i} \in P_t$: décodage: $\mathbf{x} = c^{-1}(\mathbf{i})$ calcul des objectifs: $\mathbf{f}(\mathbf{x})$
Step 2.2	$P' = \emptyset$ $i = 1$
Step 2.3	Sélection aléatoire de: $\mathbf{i}, \mathbf{j} \in P_t$ $P_{comp} \subset P_t$ ($\sharp P_{comp} = n_{comp}$)
Step 2.4	Si $\mathbf{f}(\mathbf{i})$ non dominée % $\mathbf{f}(P_{dom})$ et $\mathbf{f}(\mathbf{j})$ dominée % $\mathbf{f}(P_{dom})$ alors $P' = P' \cup \{\mathbf{i}\}$ passer à Step 3.2
Step 2.5	Si $\mathbf{f}(\mathbf{i})$ dominée % $\mathbf{f}(P_{comp})$ et $\mathbf{f}(\mathbf{j})$ non dominée % $\mathbf{f}(P_{comp})$ alors $P' = P' \cup \{\mathbf{j}\}$ passer à Step 3.2
Step 3.1	$n(\mathbf{i}) = \sharp\{\mathbf{k} \mid \mathbf{k} \in P' \wedge d(\mathbf{i}, \mathbf{k}) < \sigma_{sh}\}$ $n(\mathbf{j}) = \sharp\{\mathbf{k} \mid \mathbf{k} \in P' \wedge d(\mathbf{j}, \mathbf{k}) < \sigma_{sh}\}$ Si $n(i) \leq n(j)$ alors $P' = P' \cup \{\mathbf{i}\}$ sinon $P' = P' \cup \{\mathbf{j}\}$
Step 3.2	$i = i + 1$ Si $i \leq N_{pop}$ alors retour à Step 2.3

ALGO. 2.2 – Pseudo code de l'algorithme NPGA

2.5.3 MOGA

L'algorithme MOGA (*Multi Objective Genetic Algorithm*) a été proposé en 1993 par Fonseca et Fleming [14].

Cette technique généralise au cas multi-objectifs le schéma de sélection basé sur le classement, et s'organise comme suit (algorithme 2.3):

- A chaque génération, les individus de la population se voient attribuer un classement $r(\mathbf{i})$ défini par:

$$r(\mathbf{i}) = 1 + p(\mathbf{i}) \tag{2.29}$$

où $p(\mathbf{i})$ désigne le nombre d'individus $\mathbf{j}$ de la population qui possèdent un vecteur

Step 2.1	$\forall\ \mathbf{i} \in P_t$: décodage: $\mathbf{x} = c^{-1}(\mathbf{i})$ calcul des objectifs: $\mathbf{f}(\mathbf{x})$
Step 2.2	$\forall\ \mathbf{i} \in P_t$: calcul du classement $r(\mathbf{i}) = 1 + \sharp\{\mathbf{j} \mid \mathbf{j} \in P_t \wedge \mathbf{f}(\mathbf{j}) \text{ domine } \mathbf{f}(\mathbf{i})\}$
Step 2.3	$\forall\ \mathbf{i} \in P_t$: $f'_{ad}(\mathbf{i}) = f'_{ad}(r(\mathbf{i}))$
Step 2.4	$\forall\ \mathbf{i} \in P_t$: calcul de $f_{ad}(\mathbf{i})$: moyenne de $f'_{ad}(\mathbf{i})$ sur l'ensemble de la population
Step 2.5	$\forall\ \mathbf{i} \in P_t$ sharing de $f_{ad}(\mathbf{i})$ dans l'espace des objectifs
Step 3	$P' = \emptyset$ $\forall\ \mathbf{i} \in P_t$: choix de $\mathbf{i} \in P_t$: roue de loterie avec échantillonnage stochastique $P' = P' \cup \{\mathbf{i}\}$

ALGO. 2.3 – Pseudo code de l'algorithme MOGA

des objectifs $\mathbf{f}(\mathbf{j})$ dominant $\mathbf{f}(\mathbf{i})$.

- Pour chaque individu, une valeur intermédiaire de la fonction d'adaptation est déterminée à l'aide d'une fonction (linéaire ou non) de son classement [1]:

$$f'_{ad}(\mathbf{i}) = f'_{ad}(r(\mathbf{i})) \tag{2.30}$$

- Une moyenne des $f'_{ad}(\mathbf{i})$, effectuée sur l'ensemble de la population, permet ensuite de déterminer la valeur finale de la fonction d'adaptation de chaque individu:

$$f_{ad}(\mathbf{i}) = N_{pop}\ \frac{f'_{ad}(\mathbf{i})}{\sum_{i=1}^{N_{pop}} f'_{ad}(\mathbf{i})} \tag{2.31}$$

- Un opérateur de sharing est appliqué aux individus de même classement et ce, dans l'espace des objectifs.
- Enfin, une roue de loterie avec échantillonnage stochastique permet de réaliser le processus de sélection.

La figure 2.11(d) illustre ce mécanisme de classement dans le cas d'une minimisation de deux fonctions objectifs. On constate que les individus correspondant à des solutions non dominées ont un classement $r = 1$.

Comme l'indique Coello [4], l'algorithme MOGA est connu pour sa facilité d'implémentation et son efficacité. Cependant, l'utilisation d'un opérateur de sharing nécessite le réglage

d'un nouveau paramètre (σ_{sh}) qui influence les performances de l'algorithme. De plus, cette étape de sharing étant réalisée dans l'espace des objectifs, deux configurations distinctes, qui possèdent les mêmes valeurs de fonctions objectifs, ne pourront jamais exister simultanément dans la population.

2.5.4 NSGA

L'algorithme NSGA (*Nondominated Sorting Genetic Algorithm*) a été proposé en 1994 par Srinivas et Deb [32].

Cette technique est basée sur un processus itératif de classification des individus de la population selon différentes catégories de fronts de Pareto (algorithme 2.4):

- Les solutions non dominées sont tout d'abord identifiées, puis "rangées" dans un premier front.
- Les individus déterminés à l'étape précédente se voient ensuite attribuer une fonction d'adaptation qui vaut:
$$f_{ad} = F \tag{2.32}$$
où la valeur initiale de F est généralement donnée par:
$$F = N_{pop} \tag{2.33}$$
- Un opérateur de sharing, appliqué dans l'espace de conception, est appliqué aux individus appartenant au front en cours.
- La valeur de F est ensuite actualisée au moyen de l'expression suivante:
$$F = \min_{\mathbf{i} \in F_p} f_{ad}^{sh}(\mathbf{i}) \tag{2.34}$$
- Une fois le premier tri effectué, les solutions non dominées sont "effacées" de la population, et le processus est de nouveau appliqué à la population restante. Cette fois, les individus correspondant au nouveau front reçoivent une fonction d'adaptation définie par l'équation (2.34).
- Ce processus de classement est répété jusqu'à ce que la totalité de la population soit triée.
- Enfin, une sélection basée sur le tournoi permet de choisir les individus les mieux adaptés.

La figure 2.11(c) illustre, dans le cas d'une minimisation de deux fonctions objectifs, les différents fronts de Pareto obtenus à l'aide de cette procédure.

D'après Coello [4], NSGA semble moins efficace que MOGA (en termes de temps de calcul et de qualité des fronts de Pareto obtenus), et il apparaît aussi plus "sensible" aux valeurs du facteur σ_{sh}. Par contre, l'étape de sharing étant cette fois appliqué dans l'espace de conception, plusieurs configurations équivalentes peuvent exister simultanément dans la population.

Step 2.1	$\forall$ **i** $\in P_t$: décodage: $\mathbf{x} = c^{-1}(\mathbf{i})$ calcul des objectifs: $\mathbf{f}(\mathbf{x})$
Step 2.2	$P_{nondom} = \emptyset$ $P_r = P_t$ $F = N_{pop}$
Step 2.3	$\forall$ **i** $\in P_r$: Si **f**(**i**) est non dominée alors $P_{nondom} = P_{nondom} \cup \{\mathbf{i}\}$ $P_r = P_r \setminus \{\mathbf{i}\}$
Step 2.4	$\forall$ **i** $\in P_{nondom}$: calcul de la fonction d'adaptation: $f_{ad}(\mathbf{i}) = F$ sharing de $f_{ad}(\mathbf{i})$dans l'espace de conception
Step 2.5	$F = \min\{f_{ad}(\mathbf{i}) \mid \mathbf{i} \in P_{nondom}\}$
Step 2.6	Si $P_r \neq \emptyset$ alors retour Step 2.3
Step 3	$P' = \emptyset$ $\forall$ **i** $\in P_t$: choix de **i** $\in P_t$ selon schéma de sélection par tournoi $P' = P' \cup \{\mathbf{i}\}$

ALGO. 2.4 – Pseudo code de l'algorithme NSGA

2.6 Comparaisons des techniques a priori et a posteriori

Ce chapitre consacré aux méthodes de résolution d'un problème multi-objectifs ne se voulait pas exhaustif. Le but de ce travail étant le développement d'un algorithme génétique, nous avons principalement envisagé des méthodes pouvant être implémentées dans ce type de métaheuristique. Néanmoins, à l'aide des notions de dominance et d'optimum de Pareto, nous avons pu dégager les avantages et les inconvénients respectifs de deux grandes familles de méthodes d'optimisation multi-objectifs.

Les techniques a priori consistent à résoudre un problème mono-objectif dérivé du problème multi-objectifs initial et ce, sur base d'indications préalablement fournies par le concepteur. Selon le type d'informations données dans la phase de décision, on distingue:

- les techniques d'agrégation (méthodes des combinaisons linéaires et des distances) où

les fonctions objectifs sont "regroupées" au sein d'une unique fonction de préférence et les choix portent sur la valeur des coefficients de pondération,

- les techniques de priorité (méthodes des contraintes et de l'ordonnancement) où la fonction de préférence s'apparente à l'objectif jugé prioritaire, les autres objectifs introduisent quant à eux des contraintes supplémentaires.

La minimisation de la fonction de préférence ainsi obtenue peut ensuite être réalisée à l'aide d'une méthode classique du gradient (pour autant que cette fonction soit dérivable), ou d'une métaheuristique (algorithme génétique, recuit simulé, etc.).

Les techniques a priori souffrent cependant toutes du même inconvénient: au terme de l'étape de recherche, une seule solution optimale est calculée. L'identification d'une autre solution du front de Pareto nécessite donc une réinitialisation du problème (nouvelles valeurs de coefficients de pondération ou autres priorités des objectifs), et le processus d'optimisation doit ensuite être à nouveau effectué. De plus, si l'on ne dispose d'aucune information sur la forme du front de Pareto, une répartition homogène des points le discrétisant est très difficile à obtenir.

Les techniques a posteriori consistent à déterminer, sans information préalable, un ensemble de solutions optimales parmi lesquelles le concepteur sélectionne, à la fin du processus, celle qui le satisfait le plus. Il y a donc un gain de temps non négligeable par rapport à la phase de modélisation des préférences inhérente aux méthodes a priori.

Les algorithmes génétiques, de par leur principe de fonctionnement, s'apparentent à une technique a posteriori: à la fin du processus de recherche, ils fournissent au concepteur un ensemble de solutions non dominées. En outre, l'emploi d'un opérateur de sharing semble permettre une distribution de ces solutions de manière uniforme sur le front de Pareto.

Parmi les mises en œuvre algorithmiques étudiées, les techniques qui permettent de prendre en compte la notion de dominance de Pareto, via une modification des opérateurs d'évaluation et de sélection présents dans l'algorithme génétique "de base", semblent les plus efficaces (MOGA, NSGA et NPGA). Comme en témoigne une abondante littérature, elles sont fréquemment employées avec succès dans divers domaines d'application.
Malheureusement, comme l'indique Zitzler [39], il est très difficile de les comparer car les critères de performance sont nombreux (vitesse de convergence, distribution homogène des solutions sur le front de Pareto, nombre de solutions optimales obtenues à la fin du processus, etc.).

Chapitre 3

L'algorithme développé: GAGERO

3.1 Prise en compte des contraintes

Lorsque qu'un algorithme génétique est utilisé pour résoudre un problème d'optimisation mono-objectif sous contraintes (tel qu'il a été défini dans l'introduction), la démarche habituelle consiste à utiliser une **méthode de pénalités statiques**. Cette technique consiste à minimiser une pseudo fonction objectif:

$$F(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^{n_g} k_j \; P_j^{\beta}(\mathbf{x}) \tag{3.1}$$

où:

- k_j (≥ 0 dans le cas d'une minimisation) sont les coefficients définissant l'intensité des pénalités,
- β est un paramètre imposé par l'utilisateur (généralement fixé dans l'intervalle $[1, 2]$),
- $P(\mathbf{x})$ désigne la fonction de pénalité qui s'écrit:

$$P_j(\mathbf{x}) = \max\{0, g_j(\mathbf{x})\} \qquad j = 1, \ldots, n_g \tag{3.2}$$

Cette technique nécessite alors de fixer judicieusement les coefficients de pénalité k_j: des valeurs trop élevées conduisent généralement à une convergence prématurée de l'algorithme, alors que des valeurs trop faibles ralentissent la convergence et augmentent inutilement le temps de calcul.

La **méthode de pénalités dynamiques**, consiste quant à elle à utiliser des coefficients dont la valeur évolue au cours de l'algorithme: lors des premières générations, les intensités sont faibles, de manière à permettre une large exploration de l'espace de conception ; ces intensités sont ensuite progressivement augmentées de façon à diriger et concentrer la

population vers des zones admissibles de l'espace de recherche.
Joines et Houck [19] proposent par exemple la formulation suivante:

$$F(\mathbf{x}) = f(\mathbf{x}) + (C \cdot t)^{\alpha} \sum_{j=1}^{n_g} P_i^{\beta}(\mathbf{x}) \tag{3.3}$$

où C, α, β sont des paramètres imposés par l'utilisateur (habituellement, $C = 0.5$, $\alpha = 1$, $\beta = 2$), et t désigne le numéro de la génération en cours.

Dans le cas d'un problème d'optimisation multi-objectifs sous contraintes, l'approche traditionnelle consiste à utiliser une technique a priori (afin de se ramener à un problème mono-objectif), couplée à une méthode de pénalité.
Outre les inconvénients liés aux techniques a priori qui ont été mis en évidence dans le chapitre précédent, cette approche nécessite toujours un choix adéquat des coefficients de pénalité.

Bien que les algorithmes MOGA, NSGA et NPGA envisagés précédemment permettent une prise en compte de astucieuse de l'aspect multi-objectifs, ils n'offrent malheureusement pas la possibilité de traiter les contraintes.

Nous avons donc développé une **méthode originale** [20] qui permet de prendre en compte les contraintes d'inégalité, tout en utilisant la notion de dominance de Pareto présente dans l'algorithme MOGA. Notre algorithme, baptisé GAGERO (*Genetic Algorithm for GERotor Optimization*) s'organise donc comme suit (figure 3.1):

- Les contraintes sont tout d'abord évaluées pour chaque individus de la population. Cette étape permet de déterminer deux sous-populations: l'une composée d'individus satisfaisant toutes les contraintes, l'autre contenant les individus violant une (ou plusieurs) de ces contraintes.
- Les fonctions objectifs sont ensuite calculées pour tous les individus appartenant à la sous-population admissible. Ceux-ci reçoivent ensuite un classement selon l'algorithme MOGA.
- Les individus non admissibles se voient quant à eux attribuer un facteur de "viol" défini par:

$$R_{const} = \sum_{j=1}^{n_g} \max\{0, g_j\} \tag{3.4}$$

- Les deux sous-populations sont enfin mélangées et une sélection basée sur le tournoi pénalisé est appliquée. Ce nouveau type de tournoi consiste à choisir aléatoirement deux individus dans la population et ensuite les comparer:
  - s'ils sont tout les deux admissibles, le gagnant est celui qui possède le meilleur classement,
  - si un seul des deux individus est admissible, il remporte le tournoi,
  - s'ils sont tout les deux non admissibles, les gagnant est celui possède le facteur de viol le plus petit.

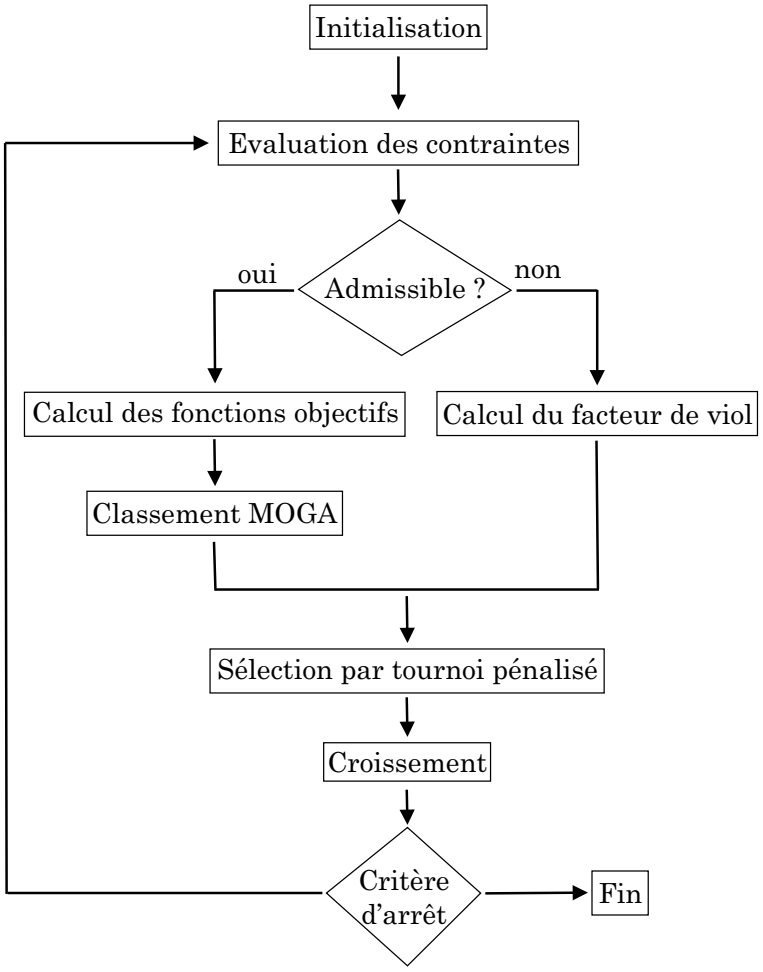


FIG. 3.1 – Prise en compte de l'aspect mutli-objectifs et contraint dans GAGERO

3.2 Les opérateurs génétiques utilisés

Au vu des avantages et des inconvénients des différentes options qui ont été mis en lumière tout au long des chapitres précédents, nous avons basé notre algorithme génétique sur:

- un codage réel des variables,
- un opérateur de croisement fonction de la nature des variables auxquelles il s'applique, soit:
  - une recombinaison linéaire de type BLX-alpha pour les variables continues,
  - un croisement mono-point pour les variables discrètes,
- un opérateur de sélection fonction du type d'optimisation, soit:
  - un classement linéaire associé à une roue de loterie avec échantillonnage stochastique pour le cas mono-objectif non contraint,
  - un classement linéaire (avec une roue de loterie ou un tournoi) couplé à MOGA pour le cas multi-objectifs non contraint

– un tournoi pénalisé couplé à MOGA pour le cas multi-objectifs contraint.

Signalons également que nous avons modifié l'opérateur de croisement BLX-alpha afin de prendre en compte les contraintes de bornes de la manière suivante:

- Chaque composante (gène) d'un individu de le population initiale est choisie aléatoirement dans son intervalle de définition. Ainsi, la totalité de la population initiale satisfait les contraintes de bornes.
- Ensuite, chacune des composantes d'un enfant créé par croisement est testée: si elle appartient bien à l'intervalle de définition, elle est conservée dans le chromosome de l'enfant ; dans le cas contraire, elle est remplacée par le gène lui correspondant chez l'un des parents, choisi aléatoirement.

Les valeurs des différents paramètres intervenant dans l'algorithme sont fournies au code via un fichier de données dans le lequel l'utilisateur spécifie:

- le nombre ainsi que le domaine de définition des variables continues et discrètes intervenant dans le problème,
- N_{pop}: la taille de la population,
- α: le taux d'exploration au-delà des parents lors du croisement des variables continues,
- S_p: la pression de sélection utilisée lors du classement linéaire,
- G_{max}: le nombre maximum de générations,
- ϵ: la précision de la convergence (uniquement dans le cas mono-objectif).

Signalons enfin que nous avons choisis le C++ (dans l'environnement *Microsoft Developper Visual C++*) comme langage de programmation car:

- Il est à la fois rapide, performant et portable.
- Le code s'articule selon une structure modulaire: il est donc très facile d'ajouter, de supprimer ou de modifier un opérateur génétique par la suite.
- Une liaison avec d'autres codes de calcul est réalisable: notre algorithme génétique peut être ainsi couplé à un logiciel prenant en charge l'évaluation de la fonction objectif. Pour preuve, lors de son application à un problème industriel qui sera détaillé dans un chapitre ultérieur, notre outil a été aisément interfacé avec un code de calcul écrit en Fortran et développé par Techspace Aero.

Chapitre 4

Application de l'algorithme à des problèmes mathématiques

4.1 Cas-tests mono-objectif

4.1.1 Différentes fonctions continues

La fonction f_1 de DeJong [16], illustrée à la figure 4.1, est multi-variables, convexe, unimodale et définie par:

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{4.1}$$

avec:

$$x_i \in [-5.12, 5.12] \qquad i = 1, \ldots, n \tag{4.2}$$

Elle présente un minimum global:

$$\min(f_1) = f_1(\mathbf{0}) = 0 \tag{4.3}$$

La fonction f_2 de DeJong [16], illustrée à la figure 4.2, est multi-variables, convexe, unimodale et définie par:

$$f_2(\mathbf{x}) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \tag{4.4}$$

avec:

$$x_i \in [-2.048, 2.048] \qquad i = 1, \ldots, n \tag{4.5}$$

Cette fonction, connue aussi sous le nom de fonction “banane” ou encore “vallée de Rosenbrock” présente un minimum global:

$$\min(f_2) = f_2(\mathbf{1}) = 0 \tag{4.6}$$

qui est localisé à l'intérieur d'une longue vallée en forme de parabole. Si la découverte de cette vallée est triviale, la localisation précise de l'optimum est nettement plus difficile.

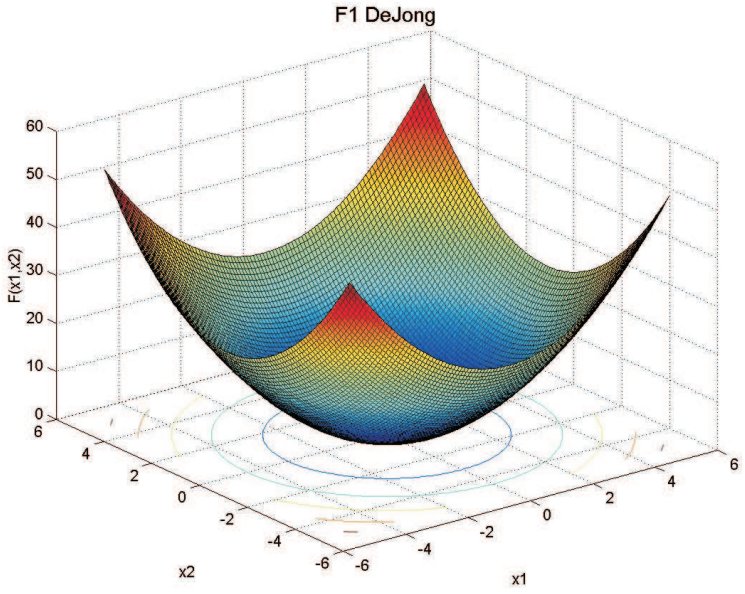


FIG. 4.1 – La fonction f_1 de DeJong (n=2)

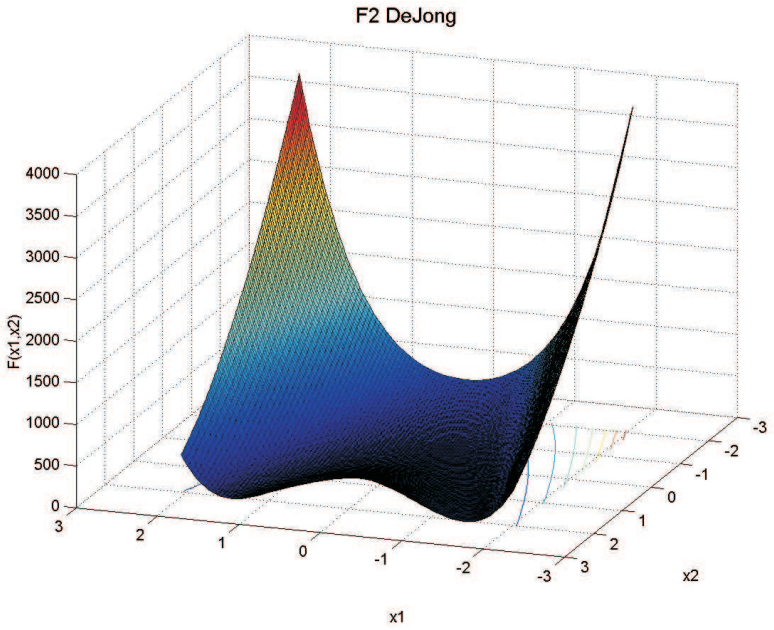


FIG. 4.2 – La fonction f_2 de DeJong (n=2)

La fonction de Rastrigin [16], illustrée à la figure 4.3, est multi-variables, convexe, multi-modale et définie par:

$$f_{Rast}(\mathbf{x}) = n + \sum_{i=1}^{n}(x_i^2 - \cos(2\pi x_i)) \tag{4.7}$$

avec:

$$x_i \in [-5.12, 5.12] \qquad i = 1, \ldots, n \tag{4.8}$$

Cette fonction est basée sur la fonction f_1 de DeJong à laquelle on a ajouté une modulation de cosinus pour produire de multiples minima locaux. Elle présente un minimum global:

$$\min(f_{Rast}) = f_{Rast}(\mathbf{0}) = 0 \tag{4.9}$$

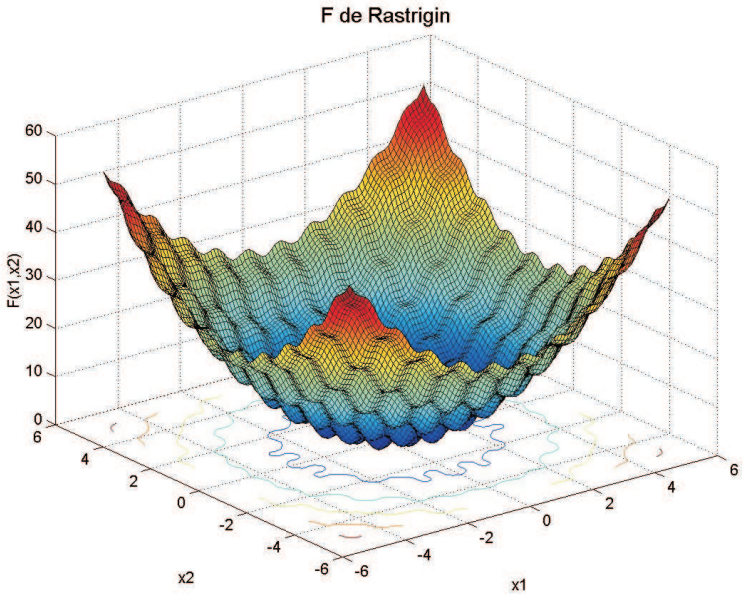


FIG. 4.3 – La fonction f_{Rast} de Rastrigin (n=2)

La fonction de Easom [8], illustrée à la figure 4.4, est multi-variables, convexe, unimodale et définie par:

$$f_{Eas}(x_1, x_2) = -\cos(x_1) \cdot \cos(x_2) \cdot e^{-\left((x_1-\pi)^2+(x_2-\pi)^2\right)} \tag{4.10}$$

avec:

$$x_i \in [-10, 10] \qquad i = 1, 2 \tag{4.11}$$

Elle présente un minimum global:

$$\min(f_{Eas}) = f_{Eas}(\pi, \pi) = -1 \tag{4.12}$$

qui est localisé dans une zone très étroite par rapport à l'espace de définition.

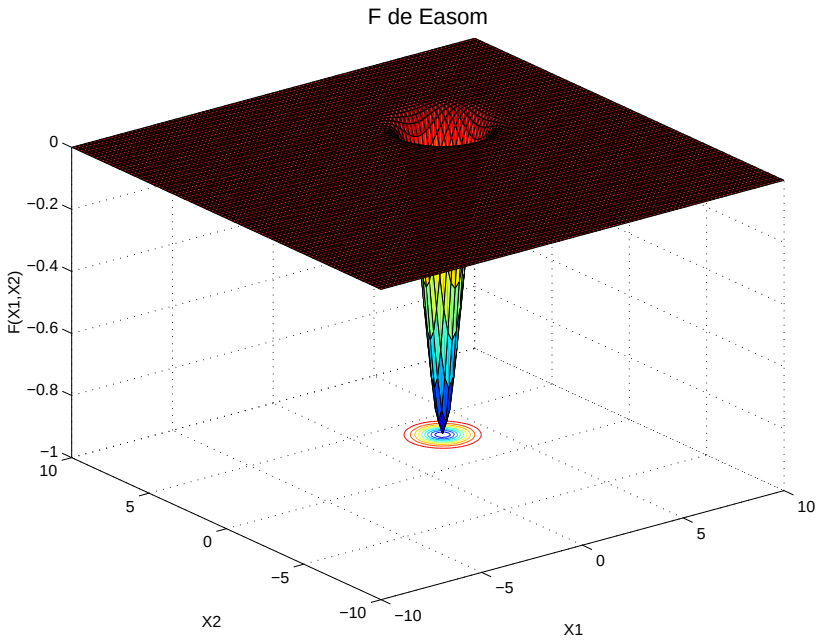


FIG. 4.4 – La fonction f_{Eas} de Easom

Résultats

Nous présentons tout d'abord les résultats relatifs à 2 variables de conception ($n = 2$ pour f_1, f_2 et f_{Rast}). Les valeurs des paramètres intervenant dans l'algorithme génétique sont précisées au tableau 4.1.

Paramètre	Valeur
N_{pop}	100
S_p	2
α	0.9
ϵ	10^{-6}
G_{max}	200

TAB. 4.1 – Paramètres employés pour les cas-tests mono-objectif

Pour juger de la convergence du code, nous avons calculé, à chaque génération considérée, les variables suivantes:

- Best et Worst qui correspondent respectivement à la valeur de la fonction objectif du "meilleur" et du "plus mauvais" élément de la population.
- Average qui donne la valeur moyenne de la fonction objectif sur l'entièreté de la population:

$$\text{Average} = \frac{\sum_k^{N_{pop}} f_{obj}(\mathbf{i}_k)}{N_{pop}} \tag{4.13}$$

Cas-test	Solution obtenue			Nombre d'itérations
	x_1	x_2	$f(x_1, x_2)$	
f_1	$1.8\ 10^{-5}$	$-1.6\ 10^{-6}$	$5.8\ 10^{-10}$	48
f_2	1	1	$2.6\ 10^{-10}$	52
f_{Rast}	$2.1\ 10^{-6}$	$7.6\ 10^{-7}$	$1.0\ 10^{-10}$	50
f_{Eas}	3.141588	3.141594	-1	65

TAB. 4.2 – Performances pour les cas-tests mono-objectif en variables continues

Comme l'indique le tableau 4.2 ainsi que les figures 4.5 à 4.8, les résultats sont probants tant sur les plans qualitatif que quantitatif.

Pour chaque cas-test, l'optimum est correctement identifié après un nombre raisonnable d'itérations, et à l'aide d'une population contenant peu d'individus ($N_{pop} = 100$).
GAGERO démontre ici sa capacité à localiser un optimum "caché" dans une vallée (fonction f_2) ou "perdu" dans une zone étroite par rapport à un large domaine de définition (fonction f_{Eas}). Il prouve en outre son aptitude à s'extraire de minima locaux pour localiser efficacement l'optimum global (fonction f_{Rast}).

Signalons que les méthodes classiques d'ordre un ou supérieur (telle que BFGS, SQP ou GCMMA) fournissent généralement l'optimum après une dizaine d'itérations dans le cas de la fonction f_1. Ces techniques présentent par contre des difficultés de convergence dans le cas des fonctions f_2 et f_{Eas}. Enfin, elles sont fréquemment piégées par un optimum local dans le cas de la fonction f_{Rast}.

Nous avons également réalisé une analyse de sensibilité des performances obtenues par rapport aux paramètres de réglage disponibles dans l'algorithme. Les résultats de cette étude sont résumés au tableau 4.3 dans le cas de la fonction-test f_2. Néanmoins, les phénomènes observés, ainsi que les conclusions données ci-dessous, restent valables pour les autres cas-tests.

S_p	α	N_{pop}	n	G_{max}	Nombre d'itérations	$f(\mathbf{x}^*)$
1.0	0.9	100	2	500	500	$8.923\ 10^{-4}$
1.4	0.9	100	2	500	500	$7.826\ 10^{-4}$
1.8	0.9	100	2	500	91	$3.012\ 10^{-11}$
2.0	0.9	100	2	500	52	$2.6\ 10^{-10}$
2.0	0.1	100	2	500	51	$2.213\ 10^{-3}$
2.0	0.5	100	2	500	42	$1.823\ 10^{-10}$
2.0	1.0	100	2	500	64	$2.879\ 10^{-11}$
2.0	2.0	100	2	500	500	$3.076\ 10^{-4}$
2.0	0.9	100	10	500	500	$2.005\ 10^{+1}$
2.0	0.9	1000	10	500	500	$3.764\ 10^{-3}$
2.0	0.9	5000	10	500	473	$9.401\ 10^{-9}$

TAB. 4.3 – Sensibilité des performances fonction des paramètres de réglage (cas-test f_2)

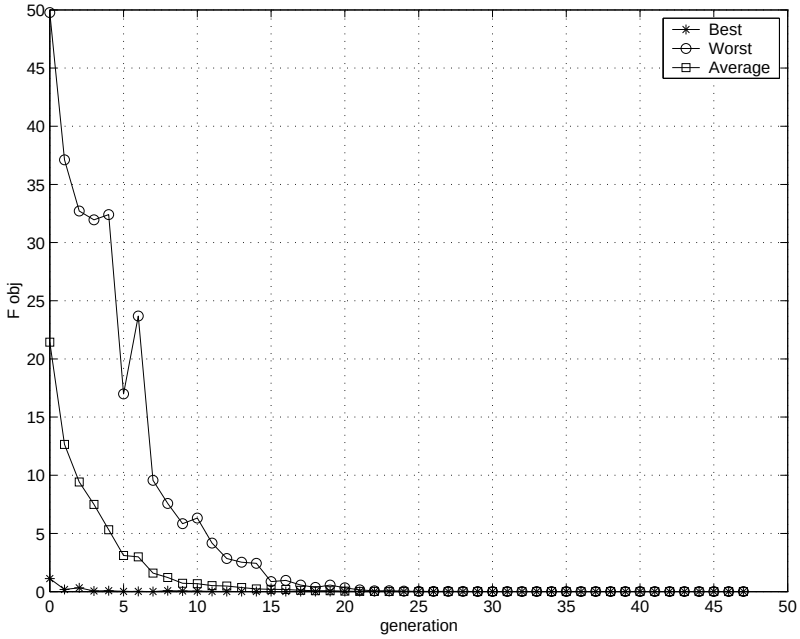


FIG. 4.5 – Courbes de convergence pour le cas-test f_1

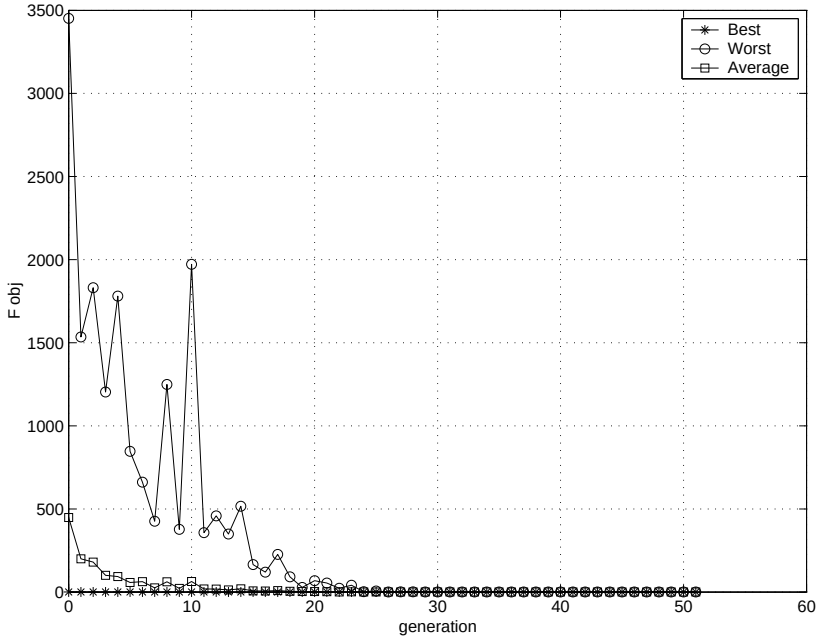


FIG. 4.6 – Courbes de convergence pour le cas-test f_2

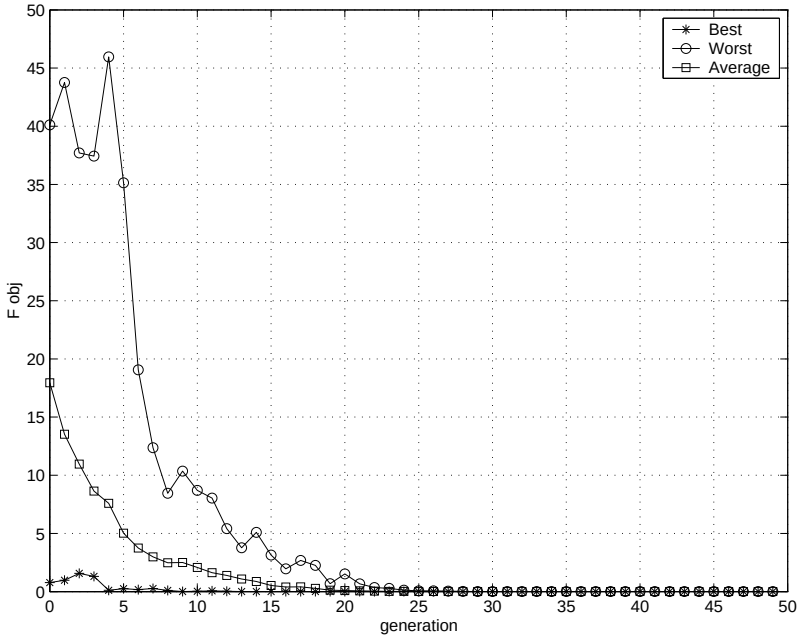


FIG. 4.7 – Courbes de convergence pour le cas-test f_{Rast}

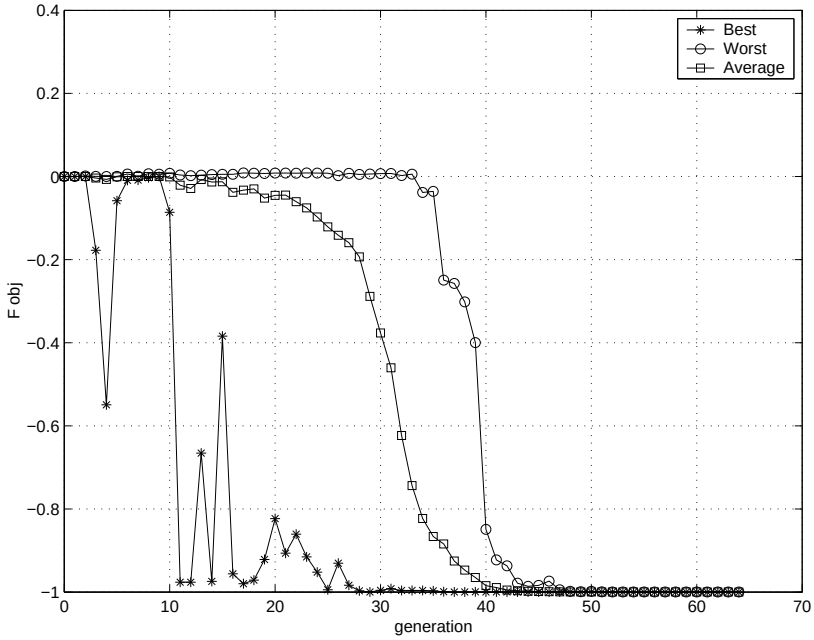


FIG. 4.8 – Courbes de convergence pour le cas-test f_{Eas}

A l'aide du tableau 4.3, on constate que le paramètre S_p (employé dans l'opérateur de sélection par classement pour régler la pression de sélection) influence à la fois la vitesse de convergence et la qualité de la solution obtenue.
En effet, une sélection trop laxiste ($S_p = 1.0$ et $S_p = 1.4$) ne permet pas d'obtenir l'optimum avec précision ($f(\mathbf{x}^*) = 8.923\ 10^{-4}$ et $f(\mathbf{x}^*) = 7.826\ 10^{-4}$). De plus, comme l'illustre la figure 4.9, l'algorithme ne semble pas converger: bien qu'une solution proche de l'optimum (courbe Best) soit conservée tout au long du processus, ni la qualité moyenne de la population (courbe Average), ni la performance du plus mauvais individu présent (courbe Worst), ne parviennent à être "uniformisées" vers cette solution.
A l'opposé, une sélection plus dure ($S_p = 1.8$ et $S_p = 2.0$) permet la localisation exacte de l'optimum ($f(\mathbf{x}^*) = 3.012\ 10^{-11}$ et $f(\mathbf{x}^*) = 2.6\ 10^{-10}$) après quelques itérations (respectivement 91 et 52).

Le paramètre α (utilisé dans le schéma de croisement BLX-alpha pour spécifier l'étendue de la zone d'exploration au-delà de celle définie par les parents) influence également la vitesse de convergence et la qualité de la solution finale.
En effet, une exploration trop limitée de l'espace de conception ($\alpha = 0.1$) ne permet pas d'obtenir l'optimum avec précision ($f(\mathbf{x}^*) = 2.213\ 10^{-3}$). Par contre, une exploration trop large de cet espace ($\alpha = 2.0$) n'offre pas la possibilité à l'algorithme de se concentrer sur les zones prometteuses, empêchant ainsi la convergence, et conduisant à une solution de moindre qualité ($f(\mathbf{x}^*) = 3.076\ 10^{-4}$).
Pour ce type de problème, il semble donc qu'une exploration raisonnable du domaine de recherche ($\alpha = 0.5$ et $\alpha = 1.0$) permette une convergence rapide de l'algorithme (respectivement 42 et 64 itérations) vers une solution optimale ($f(\mathbf{x}^*) = 1.823\ 10^{-10}$ et $f(\mathbf{x}^*) = 2.879\ 10^{-11}$).

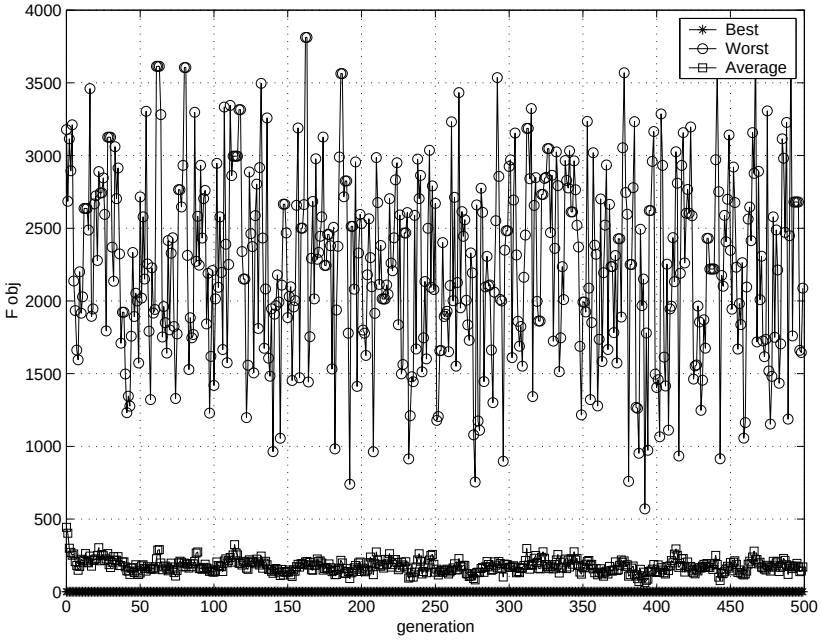


FIG. 4.9 – Courbes de convergence pour f_2 avec $S_p = 1.4$ et $\alpha = 0.9$

Enfin, pour faire face à une augmentation de la taille de l'espace de conception ($n = 10$), le nombre d'individus présents dans la population doit lui aussi augmenter. Il faut en effet atteindre une taille de population suffisante ($N_{pop} = 5000$) afin de couvrir efficacement la totalité du domaine de recherche, et ainsi trouver l'optimum ($f(\mathbf{x}^*) = 9.401\ 10^{-9}$) après un nombre relativement élevé d'itérations (473).

4.1.2 Une fonction en variables mixtes

Nous proposons ici une fonction en variables continues et discrètes:

$$f_{mixt}(\mathbf{x}) = \sum_{i_{cont}=1}^{n_{cont}} x_{i_{cont}}^2 + \sum_{i_{disc}=1}^{n_{disc}} |x_{i_{disc}}| \tag{4.14}$$

avec:

$$x_{i_{cont}} \in [-5, 5] \qquad i_{cont} = 1, \ldots, n_{cont} \tag{4.15}$$

$$x_{i_{disc}} \in \{-3, -2, -1, 0, 1, 2, 3\} \qquad i_{disc} = 1, \ldots, n_{disc} \tag{4.16}$$

Elle présente un minimum global:

$$\min(f_{mixt}) = f_{mixt}(\mathbf{0}) = 0 \tag{4.17}$$

Résultats

Les valeurs des paramètres intervenant dans l'algorithme génétique sont identiques à celles utilisées précédemment (tableau 4.1), à l'exception de la taille de la population qui a été ici adaptée en fonction du nombre de variables discrètes envisagées. Comme pour les cas-tests précédents, nous avons considéré 2 variables continues ($n_{cont} = 2$).

Comme l'indique le tableau 4.4 ainsi que la figure 4.10, l'optimum a été correctement identifié après un nombre raisonnable d'itérations.
GAGERO démontre ici sa capacité à prendre en compte efficacement l'aspect mixte des variables d'optimisation.

Solution obtenue				n_{disc}	N_{pop}	Nombre d'itérations
$x_{1,cont}$	$x_{2,cont}$	$\mathbf{x_{disc}}$	$f(\mathbf{x_{cont}}, \mathbf{x_{disc}})$			
$-1.3\ 10^{-7}$	$-2.6\ 10^{-7}$	**0**	$8.5\ 10^{-14}$	2	100	55
$-1.6\ 10^{-7}$	$-8.8\ 10^{-8}$	**0**	$3.3\ 10^{-14}$	3	100	58
$8.4\ 10^{-8}$	$1.8\ 10^{-7}$	**0**	$4.1\ 10^{-14}$	4	500	67
$5.7\ 10^{-9}$	$1.6\ 10^{-8}$	**0**	$2.8\ 10^{-16}$	5	4000	67

TAB. 4.4 – Performances pour le cas-test mono-objectif en variables mixtes

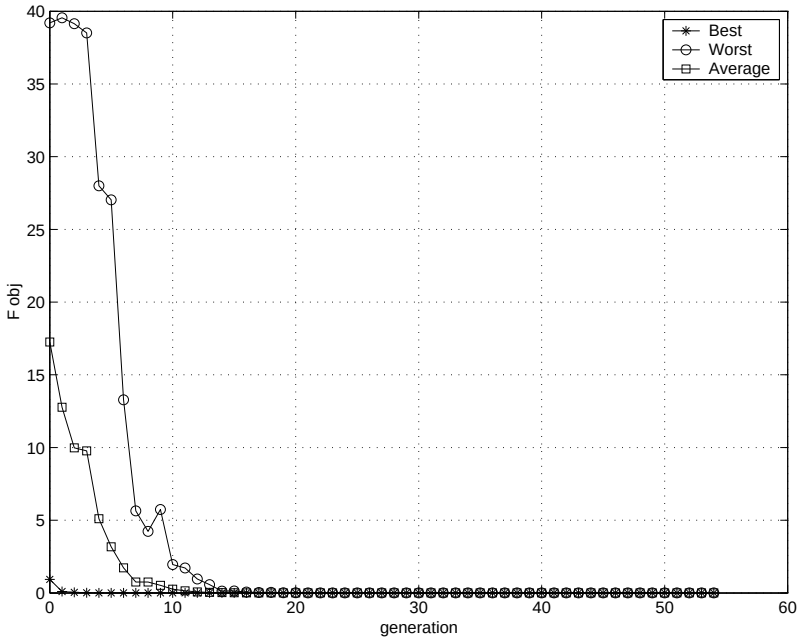


FIG. 4.10 – Courbes de convergence pour le cas-test f_{mixt} ($n_{cont} = 2$, $n_{disc} = 2$)

4.2 Cas-tests multi-objectifs

4.2.1 Les fonctions $\mathbf{T_1}$, $\mathbf{T_2}$ et $\mathbf{T_3}$ de Zitzler et Deb

Zitzler et Deb [40] proposent différents problèmes d'optimisation bi-objectifs formalisés de la manière suivante:

$$\text{minimiser} \quad \mathbf{T}(\mathbf{x}) = (f_1(x_1), f_2(\mathbf{x})) \tag{4.18}$$

$$\text{avec} \quad f_2(\mathbf{x}) = g(x_2, \ldots, x_m)\ h(f_1, g) \tag{4.19}$$

$$\text{où} \quad \mathbf{x} = (x_1, \ldots, x_m) \tag{4.20}$$

Les fonctions envisagées permettent de générer des fronts de Pareto de complexités diverses.

La fonction $\mathbf{T_1}$ est définie par:

$$f_1(x_1) = x_1 \tag{4.21}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \sum_{i=2}^{m} \frac{x_i}{m-1} \tag{4.22}$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \tag{4.23}$$

où:

$$m \in [2, 30] \tag{4.24}$$

et

$$x_i \in [0, 1] \tag{4.25}$$

Elle présente un front de Pareto continu, convexe et déterminé par:

$$g(\mathbf{x}) = 1 \tag{4.26}$$

La fonction $\mathbf{T_2}$ est définie par:

$$f_1(x_1) = x_1 \tag{4.27}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \sum_{i=2}^{m} \frac{x_i}{m-1} \tag{4.28}$$

$$h(f_1, g) = 1 - \left(\frac{f}{g}\right)^2 \tag{4.29}$$

où:

$$m \in [2, 30] \tag{4.30}$$

et

$$x_i \in [0, 1] \tag{4.31}$$

Elle présente un front de Pareto continu, non convexe et déterminé par:

$$g(\mathbf{x}) = 1 \tag{4.32}$$

La fonction $\mathbf{T_3}$ est définie par:

$$f_1(x_1) = x_1 \tag{4.33}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \sum_{i=2}^{m} \frac{x_i}{m-1} \tag{4.34}$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f}{g}\right) \sin(10\pi f_1) \tag{4.35}$$

où:

$$m \in [2, 30] \tag{4.36}$$

et

$$x_i \in [0, 1] \tag{4.37}$$

Elle présente un front de Pareto discontinu, convexe, et déterminé par:

$$g(\mathbf{x}) = 1 \tag{4.38}$$

Résultats

Afin de faciliter la visualisation, nous ne présentons ici que les résultats relatifs à 2 variables de conception ($m = 2$ pour T_1, T_2 et T_3). Les valeurs des paramètres intervenant dans l'algorithme génétique sont précisées au tableau 4.5.

Paramètre	Valeur
N_{pop}	1000
S_p	2
α	0.5
G_{max}	250

TAB. 4.5 – Paramètres employés pour les cas-tests T_1, T_2 et T_3

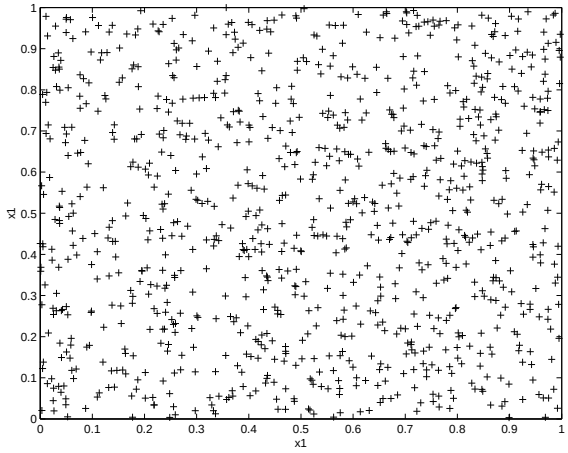
Comme l'indique le tableau 4.6 ainsi que les figures 4.11, 4.12 et 4.13, les résultats sont probants tant sur les plans qualitatif que quantitatif.
GAGERO démontre ici sa capacité à résoudre un problème d'optimisation multi-objectifs. Il prouve en outre son aptitude à discrétiser un front de Pareto convexe, concave ou discontinu.
Signalons également que les temps de calcul annoncés dans la suite de ce travail ont tous été obtenus sur un PC (Pentium III - 600 MHz - 256 Mb).

Pour ces 3 cas-tests, on constate clairement une migration de la population initiale vers les zones de l'espace de conception qui correspondent à des solutions optimales (figures 4.11(c), 4.12(c) et 4.13(c)).
Pour les cas-tests T_1 et T_2, on observe une discrétisation homogène du front de Pareto respectivement convexe et concave (figures 4.11(f) et 4.12(f)).
Dans le cas-test T_3, les différentes portions discontinues du front de Pareto (figure 4.13(f)) sont identifiées, mais la distribution des solutions non dominées est moins uniforme.

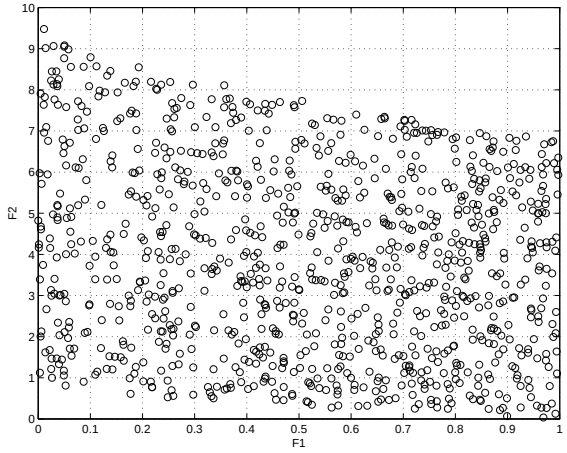
Cas-test	Nombre de points $\in F_p$	Temps de calcul (sec.)
T_1	64	66
T_2	36	64
T_3	37	66

TAB. 4.6 – Performances obtenues pour les cas-tests T_1, T_2 et T_3 (sélection par classement)

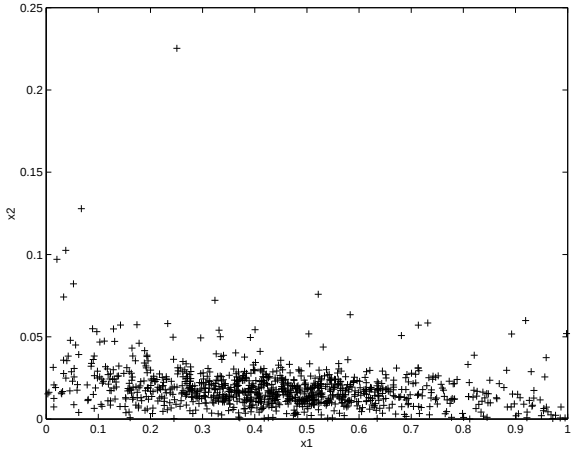
Néanmoins, dans chacun de ces cas-tests, le nombre d'individus discrétisant le front de Pareto est relativement faible: respectivement 64, 36 et 37 pour T_1, T_2 et T_3. Partant de ce constat, nous avons alors utilisé un schéma de sélection non plus basé sur le classement linéaire couplé à une roue de loterie avec échantillonnage stochastique (équation 1.10), mais basé sur un tournoi classique mettant tour à tour deux individus en compétition directe. Par conséquent, les paramètres employés dans l'algorithme génétique sont identiques à ceux donnés au tableau 4.5, à l'exception du paramètre de pression de sélection ($S_p = 2$) qui est alors remplacé par la taille du tournoi ($N_{tour} = 2$).



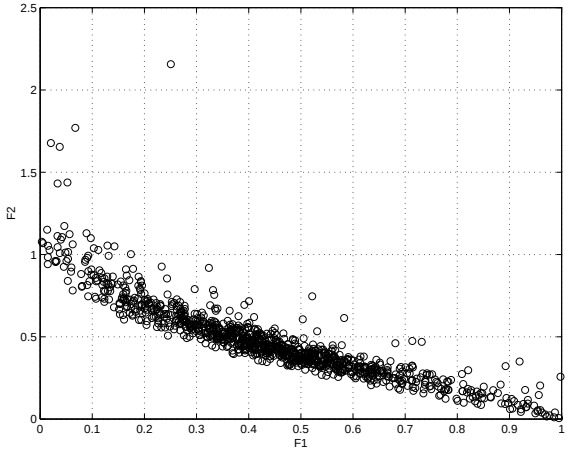
(a) espace de conception - population initiale



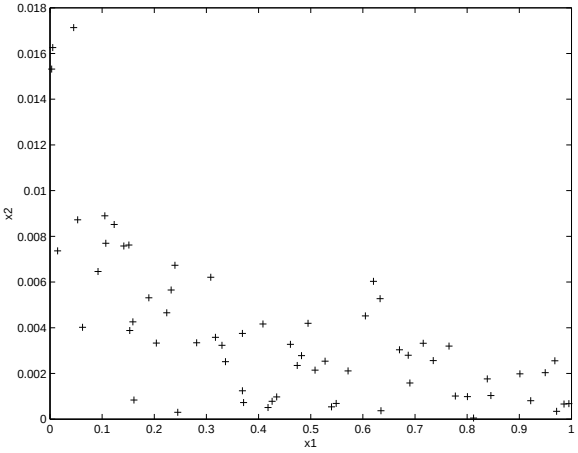
(b) espace des objectifs - population initiale



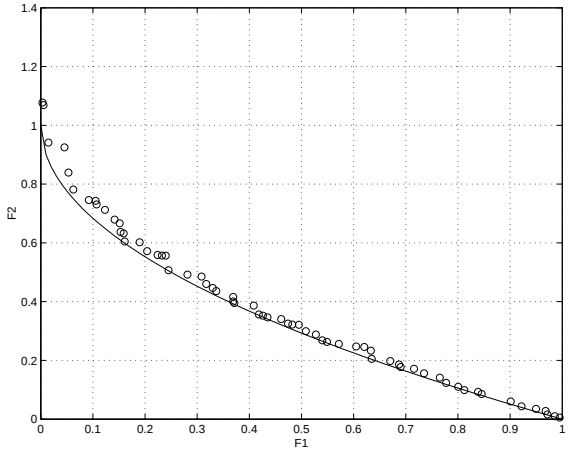
(c) espace de conception - population finale



(d) espace des objectifs - population finale

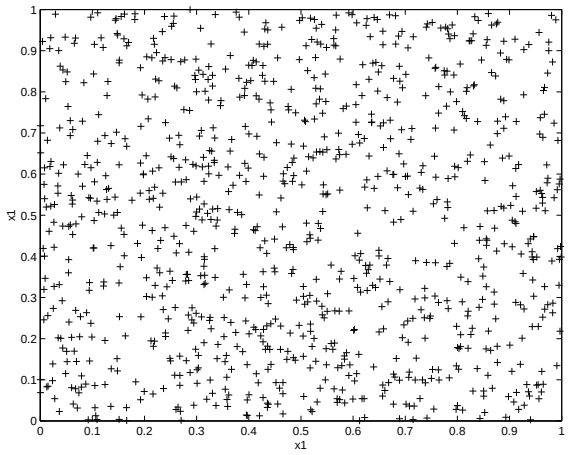


(e) espace de conception - solutions optimales

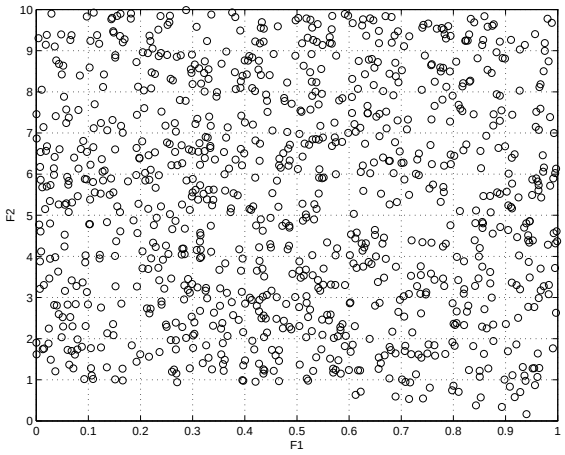


(f) espace de objectifs - solutions optimales

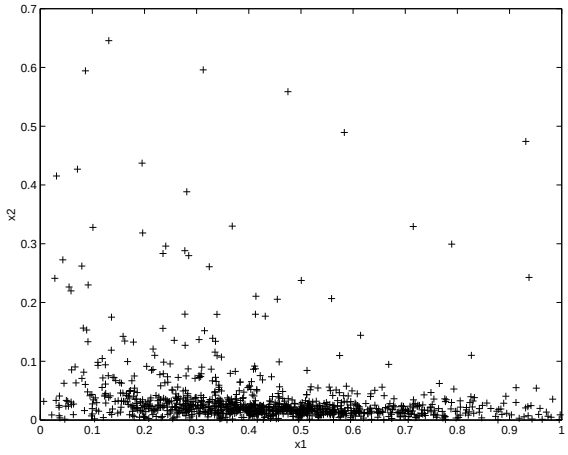
FIG. 4.11 – Évolution de la population pour le cas-test T_1 (sélection par classement)



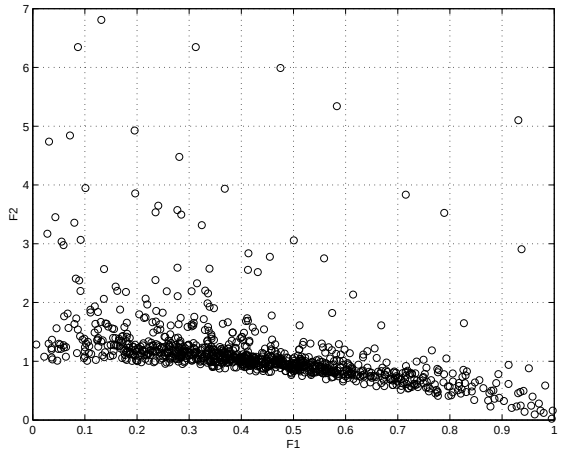
(a) espace de conception - population initiale



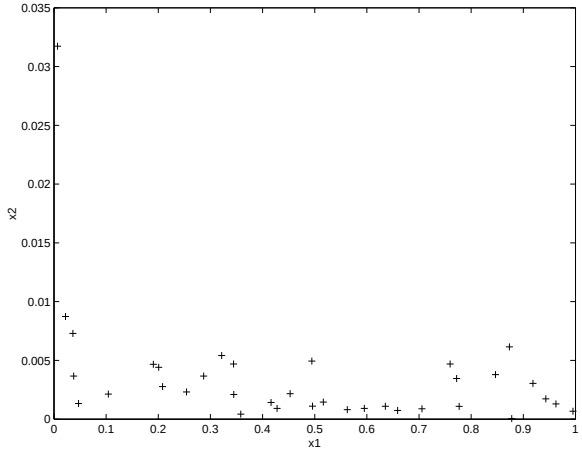
(b) espace des objectifs - population initiale



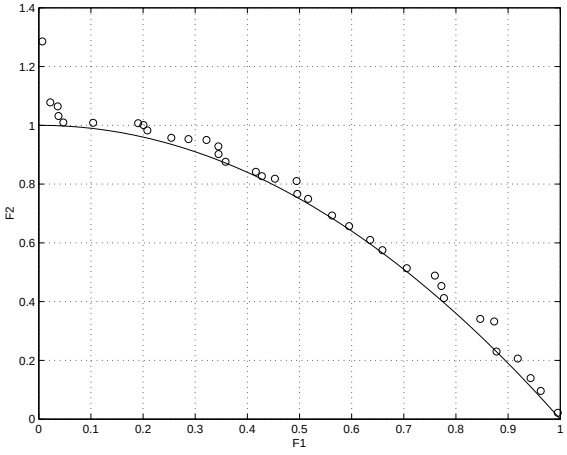
(c) espace de conception - population finale



(d) espace des objectifs - population finale

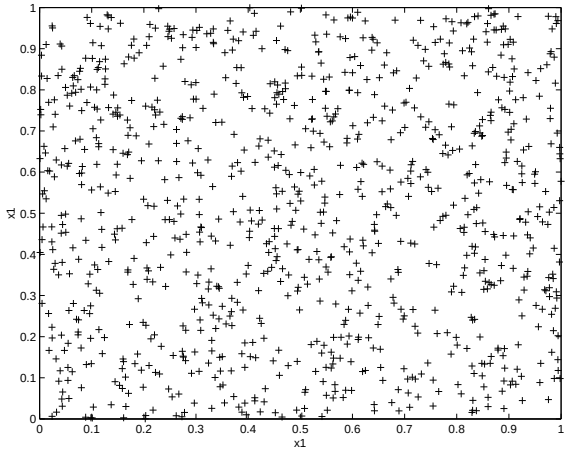


(e) espace de conception - solutions optimales

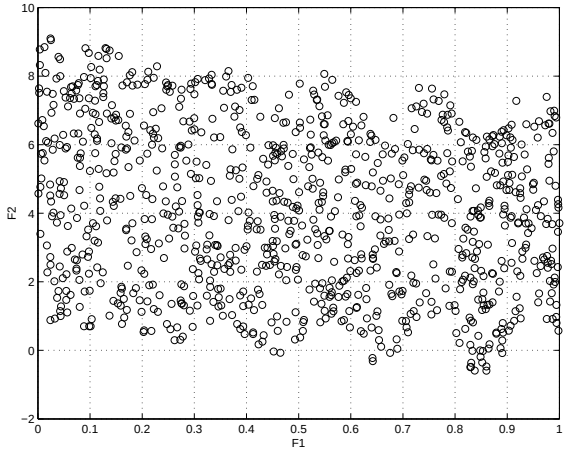


(f) espace de objectifs - solutions optimales

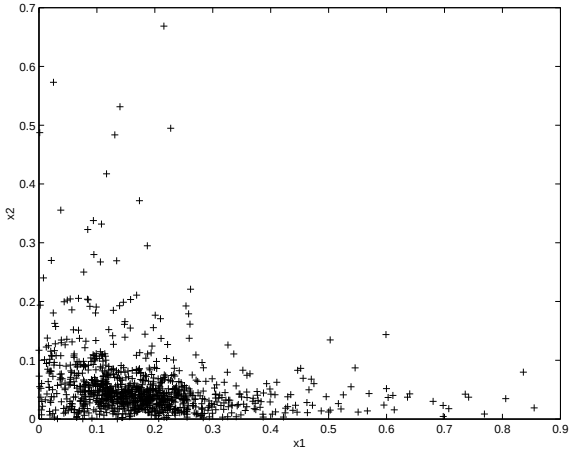
FIG. 4.12 – Évolution de la population pour le cas-test T_2 (sélection par classement)



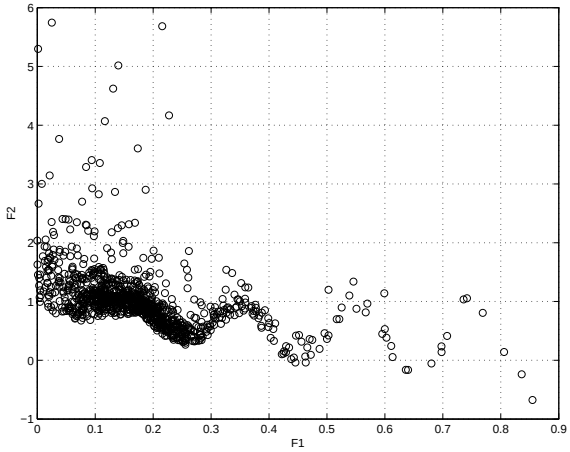
(a) espace de conception - population initiale



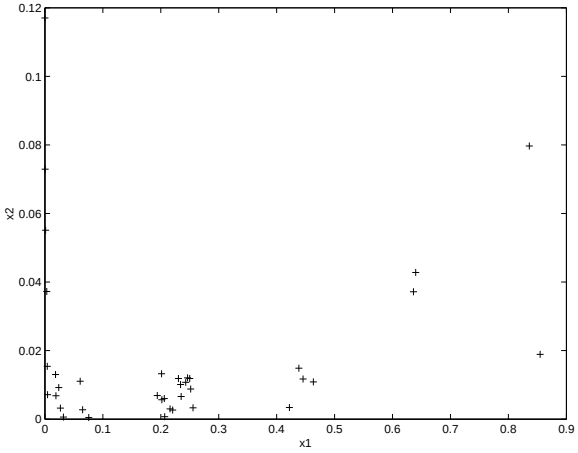
(b) espace des objectifs - population initiale



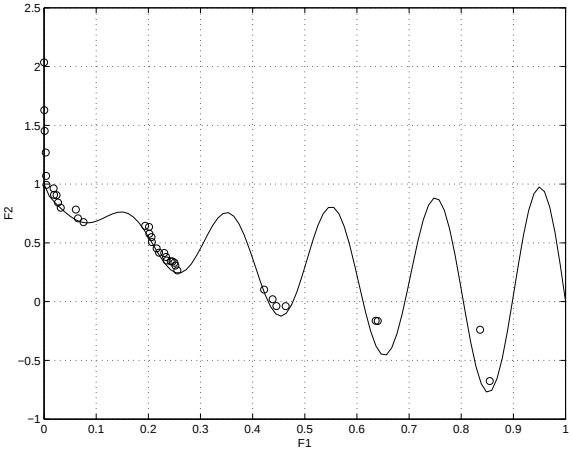
(c) espace de conception - population finale



(d) espace des objectifs - population finale



(e) espace de conception - solutions optimales



(f) espace de objectifs - solutions optimales

FIG. 4.13 – Évolution de la population pour le cas-test T_3 (sélection par classement)

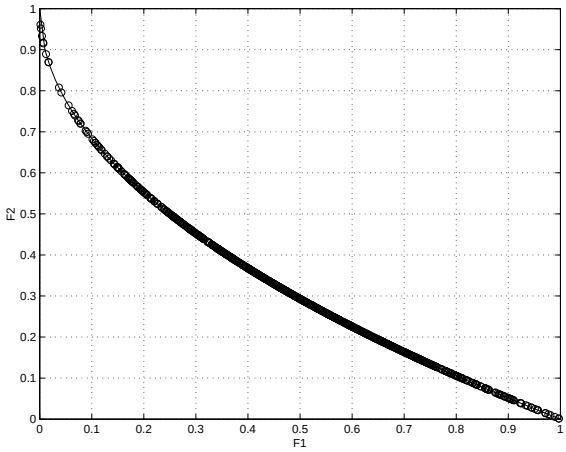


FIG. 4.14 – Front de Pareto pour le cas-test T_1 (sélection par tournoi)

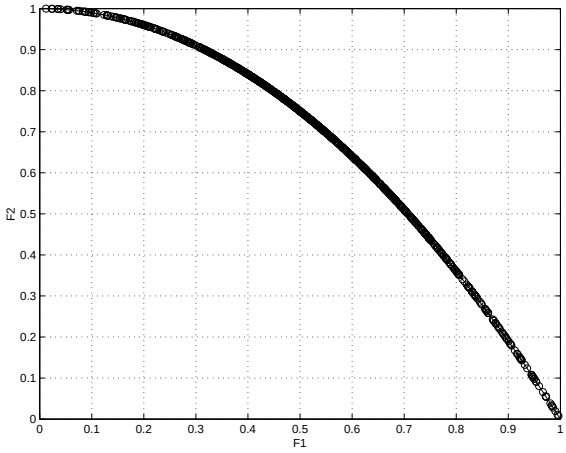


FIG. 4.15 – Front de Pareto pour le cas-test T_2 (sélection par tournoi)

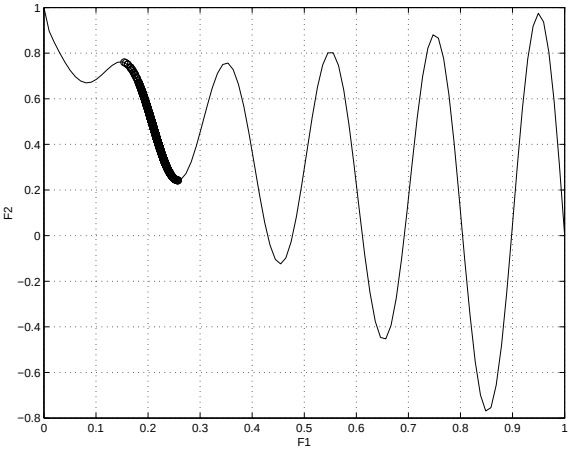


FIG. 4.16 – Front de Pareto pour le cas-test T_3 (sélection par tournoi)

Comme l'indique le tableau 4.7 ainsi que les figures 4.14, 4.15 et 4.16, les résultats sont nettement meilleurs en ce qui concerne le nombre de points appartenant au front de Pareto et ce, pour des temps de calcul semblables à ceux obtenus précédemment. En effet, on observe respectivement la présence de 956, 968 et 967 points (sur les 1000 individus que compte la population) sur le front de Pareto pour les cas-tests T_1, T_2 et T_3.

Cas-test	Nombre de points $\in F_p$	Temps de calcul (sec.)
T_1	956	70
T_2	968	73
T_3	967	74

TAB. 4.7 – Performances obtenues pour les cas-tests T_1, T_2 et T_3 (sélection par tournoi)

Pour des problèmes multi-objectifs, il semble donc qu'une sélection basée sur le tournoi permette une discrétisation plus fine du front de Pareto. Néanmoins, si la distribution des solutions non dominées est parfaitement homogène pour les cas-tests T_1 (figure 4.14) et T_2 (figure 4.15), on constate par contre que les différentes portions discontinues du front de Pareto dans le cas-test T_3 ne sont plus ici clairement identifiées (figure 4.16).

4.2.2 La fonction de Poloni

Poloni [6] propose un problème d'optimisation bi-objectifs, non contraint, formalisé de la manière suivante:

$$\text{maximiser} \quad \mathbf{P}(\mathbf{x}) = (f_1(x_1, x_2), f_2(x_1, x_2)) \tag{4.39}$$

$$\text{avec} \quad f_1(x_1, x_2) = -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2] \tag{4.40}$$

$$f_2(x_1, x_2) = -[1 + (x_1 + 3)^2 + (x_2 + 1)^2] \tag{4.41}$$

$$\text{où} \quad x_i \in [-\pi, +\pi] \tag{4.42}$$

$$A_i = \sum_{j=1}^{2} a_{ij} \ \sin(\alpha_j) + b_{ij} \ \cos(\alpha_j) \tag{4.43}$$

$$B_i = \sum_{j=1}^{2} a_{ij} \ \sin(\beta_j) + b_{ij} \ \cos(\beta_j) \tag{4.44}$$

$$\mathbf{a} = \begin{bmatrix} 0.5 & 1.0 \\ 1.5 & 2.0 \end{bmatrix} \tag{4.45}$$

$$\mathbf{b} = \begin{bmatrix} -2.0 & -1.5 \\ -1.0 & -0.5 \end{bmatrix} \tag{4.46}$$

$$\alpha = \begin{bmatrix} 1.0 & 2.0 \end{bmatrix} \tag{4.47}$$

$$\beta = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \tag{4.48}$$

Poloni propose également un second cas-test [6], basé sur la même définition des fonctions objectifs et des variables ; ces dernières doivent cependant ici satisfaire les contraintes suivantes:

$$(x_1 - 2)^2 + (x_2 - 2)^2 \leq 9 \tag{4.49}$$

$$(x_1 + 2)^2 + (x_2 + 2)^2 \geq 9 \tag{4.50}$$

Résultats

Les valeurs des paramètres intervenant dans l'algorithme génétique sont précisées au tableau 4.8. Afin de pouvoir comparer nos résultats, nous travaillons ici avec une taille de population identique ($N_{pop} = 576$) à celle employée par Bäck [6]. De plus, ayant précédemment mis en évidence l'efficacité d'une sélection basée sur le tournoi, nous continuons à employer ce schéma pour le cas-test non contraint. Pour le second cas-test, les contraintes sont prises en compte à l'aide du tournoi pénalisé intervenant dans la méthode originale que nous avons proposée.

Paramètre	GAGERO	Bäck
N_{pop}	576	576
N_{tour}	2	–
α	15	–
G_{max}	250	7500

TAB. 4.8 – Paramètres employés pour les cas-tests de Poloni

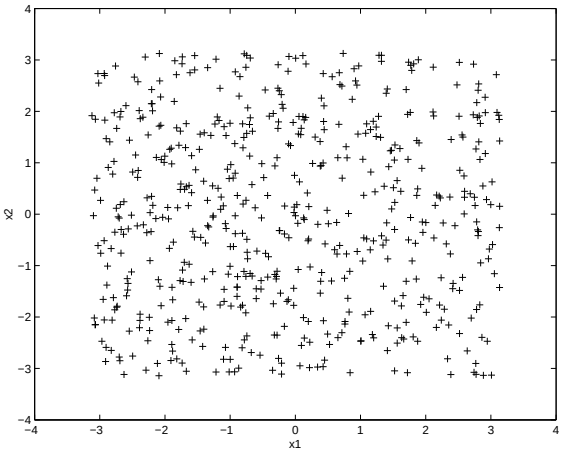
Comme l'indique le tableau 4.9 ainsi que les figures 4.17(f) et 4.18(f), on obtient à nouveau une excellente discrétisation des fronts de Pareto. En effet, on constate que respectivement 249 et 314 points ont convergé dans le cas non contraint et contraint et ce, après 250 itérations. De plus, ces performances sont meilleures que celles de Bäck qui a obtenu respectivement 90 et 114 points après 7500 itérations.

GAGERO démontre ici sa capacité à prendre en compte simultanément l'aspect multi-objectifs et contraint d'un problème d'optimisation.

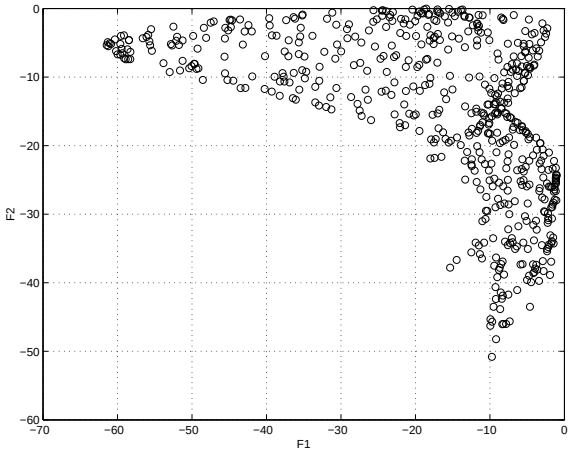
Signalons toute fois que ces cas-tests ont nécessité un réglage adéquat du paramètre de croisement ($\alpha = 15$). En effet, des valeurs plus petites pour ce paramètre ne permettent pas de capturer les portions discontinues du front de Pareto.

Performance	Poloni non contraint		Poloni contraint	
	GAGERO	Bäck	GAGERO	Bäck
nombre de points $\in F_p$	249	90	314	114
temps de calcul (sec.)	22	-	16	-

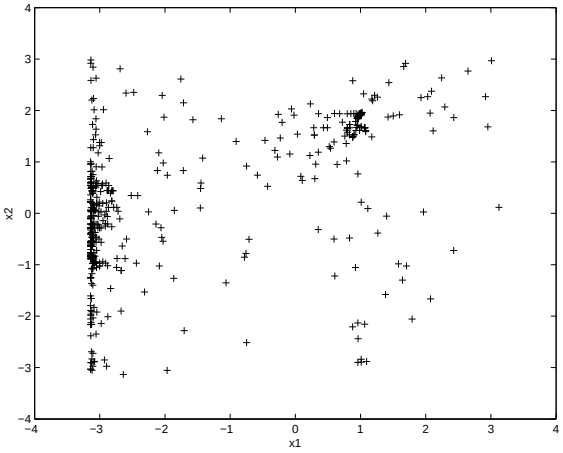
TAB. 4.9 – Performances obtenues pour les cas-tests de Poloni



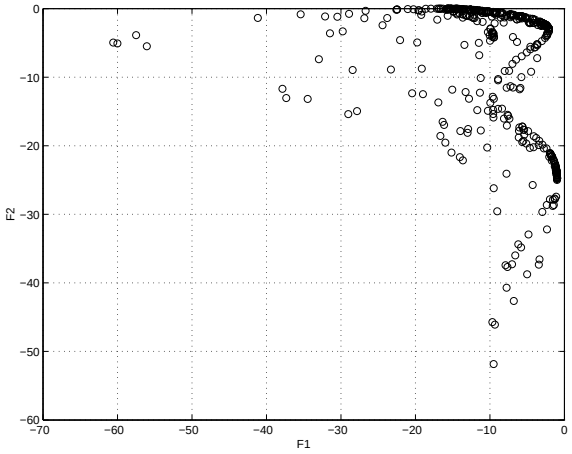
(a) espace de conception - population initiale



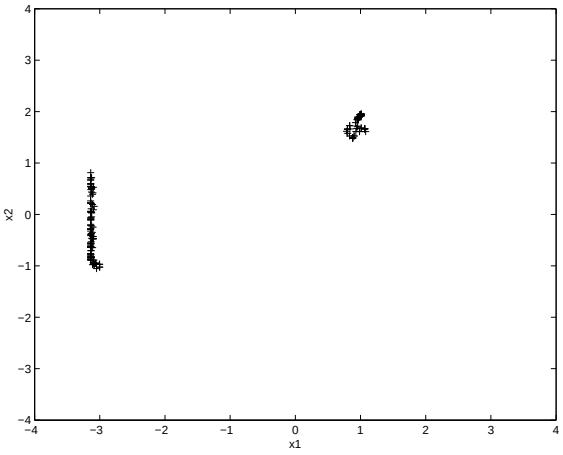
(b) espace des objectifs - population initiale



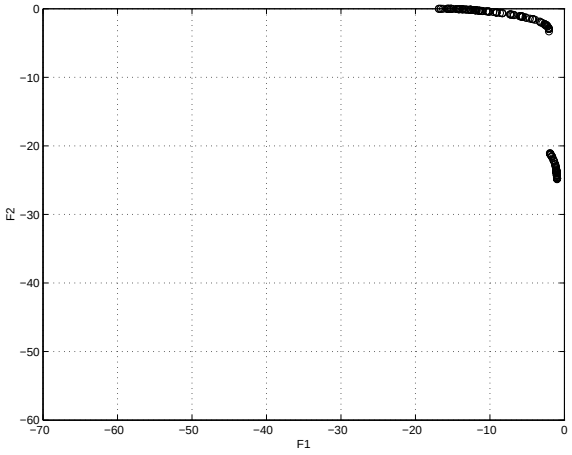
(c) espace de conception - population finale



(d) espace des objectifs - population finale

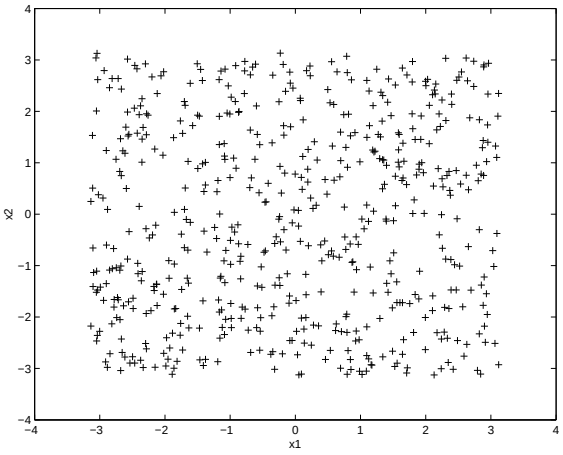


(e) espace de conception - solutions optimales

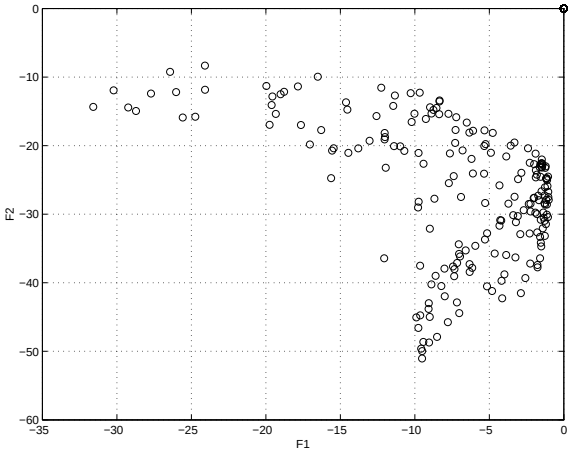


(f) espace de objectifs - solutions optimales

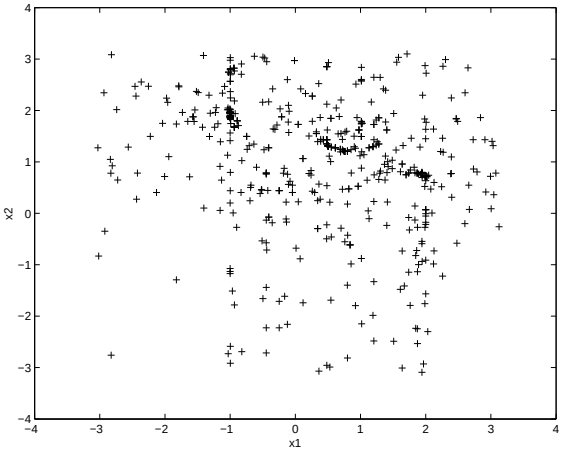
FIG. 4.17 – Évolution de la population pour le cas-test Poloni non contraint



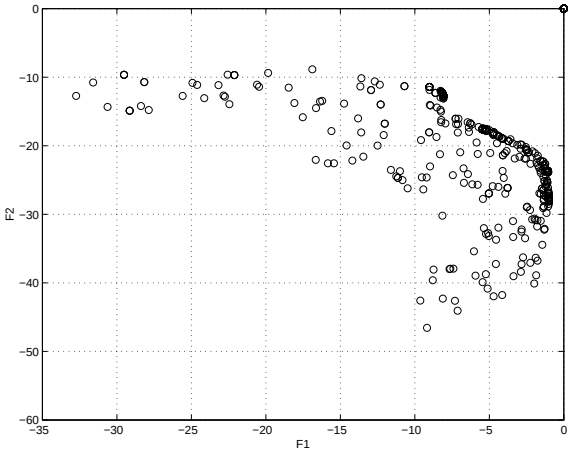
(a) espace de conception - population initiale



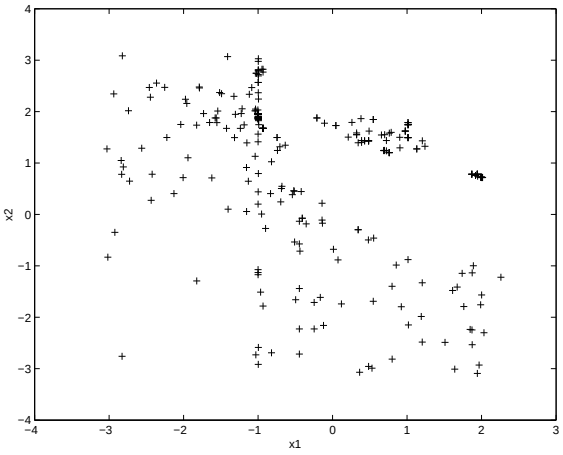
(b) espace des objectifs - population initiale



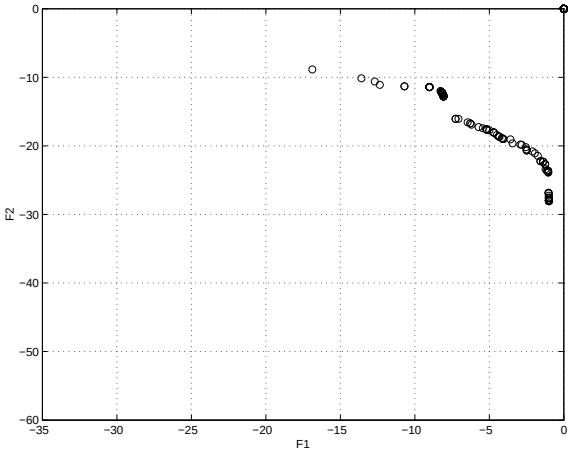
(c) espace de conception - population finale



(d) espace des objectifs - population finale



(e) espace de conception - solutions optimales



(f) espace de objectifs - solutions optimales

FIG. 4.18 – Évolution de la population pour le cas-test Poloni contraint

Chapitre 5

Application de l'algorithme à un problème industriel

5.1 Introduction

La conception et l'optimisation d'un empilage de pompes de lubrification, tel qu'il se pose à Techspace Aero, est un problème crucial pour présenter aux clients, dans les délais les plus brefs, un produit idéalement adapté à leurs besoins spécifiques.

Le dimensionnement consiste à associer plusieurs pompes, en parallèle et dans un seul carter, pour réaliser des débits spécifiés, sur un nombre variable de circuits et pour un ensemble de conditions de fonctionnement. Le problème est rendu plus ardu par le fait qu'au sein d'un groupe de lubrification, les pompes n'ont pas toutes la même fonction. Il faut donc choisir, pour chaque fonction, le nombre et le type de pompes, les dimensionner, fixer la vitesse de rotation commune, de façon à garantir non seulement tous les débits spécifiés, dans toutes les conditions de fonctionnement, mais aussi de réaliser en associant ces pompes l'ensemble le plus compact, léger et économique possible.

La difficulté de ce problème d'optimisation provient de la multiplicité des choix possibles, de leurs combinaisons, de la présence de variables de conception discrètes et continues, des nombreux objectifs (souvent contradictoires) à satisfaire, ainsi que de la forme des courbes caractéristiques des pompes.

5.2 Définition d'un empilage de gerotor

Un gerotor est une pompe volumétrique constituée d'un rotor intérieur et extérieur (figure 5.1). Le mouvement de rotation et le profil particulier des éléments dentés placés sur des

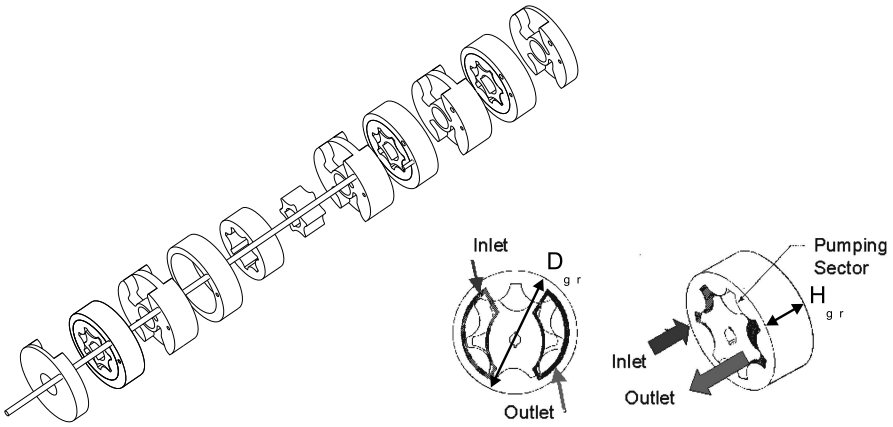


FIG. 5.1 – Un empilage de pompes gerotor

centres fixes mais excentrés l'un par rapport à l'autre engendrent des cavités plus ou moins étanches dont le volume évolue au cours de la rotation.

L'une après l'autre, chaque cavité se met en contact avec la lumière d'admission et se remplit d'huile. Au fur et à mesure de la rotation, le volume de la cavité diminue progressivement et l'huile est finalement évacuée lorsque la cavité est en contact avec la lumière de refoulement du flasque.

Dans un circuit de lubrification aéronautique, les gerotor sont utilisés d'une part comme pompe d'alimentation (qui assurent la distribution de l'huile dans les différentes enceintes à lubrifier), et d'autre part comme pompes de récupération (qui se chargent de collecter l'huile provenant des enceintes lubrifiées). Ces pompes, montées en parallèle sur un même arbre, doivent fournir des débits spécifiés pour des conditions de vols données.

Dans cette étude, nous considérons n_p pompes et n_{fc} cas de vols. De plus, la pompe indicée j dans l'empilage est définie par les paramètres suivants: l'épaisseur H_{gr}^j, le diamètre D_{gr}^j, le nombre de dents du rotor intérieur N_t^j, et la multiplicité M_{gr}^j du gerotor. Ce dernier paramètre traduit le fait que chaque fonction de pompe peut être constituée d'un sous empilage formé de gerotor identiques. Enfin, pour le cas de vol indicé i, la vitesse de rotation de l'empilage est notée N_a^i.

Un empilage de gerotor est donc défini à l'aide des vecteurs suivants:

$$\mathbf{H_{gr}} = \left[H_{gr}^1, \ldots, H_{gr}^{n_p}\right]^T \tag{5.1}$$

$$\mathbf{D_{gr}} = \left[D_{gr}^1, \ldots, D_{gr}^{n_p}\right]^T \tag{5.2}$$

$$\mathbf{N_t} = \left[N_t^1, \ldots, N_t^{n_p}\right]^T \tag{5.3}$$

$$\mathbf{M_{gr}} = \left[M_{gr}^1, \ldots, M_{gr}^{n_p}\right]^T \tag{5.4}$$

$$\mathbf{N_a} = \left[N_a^1, \ldots, N_a^{n_{fc}}\right]^T \tag{5.5}$$

Vu les phénomènes de cavitation et de pertes par fuites présents dans ce type de pompe, le débit d'huile d'un gerotor est une fonction hautement non linéaire. Dépendant des conditions de fonctionnement, des propriétés de l'huile, et de la géométrie, ce débit d'huile s'exprime en toute généralité par la relation suivante:

$$q_{gr}^{i,j} = q_{gr}^{i,j}\left(P_{in}^{i,j}, P_{out}^{i,j}, T_h^{i,j}, \rho_h^{i,j}(T_h^{i,j}), \nu_h^{i,j}(T_h^{i,j}), N_t^j, D_{gr}^j, H_{gr}^j, N_a^i\right) \tag{5.6}$$

où P_{in} and P_{out} désignent respectivement la pression d'entrée et de sortie, et T_h, ρ_h, ν_h représentent la température, la densité et la viscosité de l'huile.
L'évaluation de ce débit est réalisée par un code de calcul développé par Techspace Aero.

5.3 Formalisme mathématique du dimensionnement optimal d'un empilage

Dans ce problème de dimensionnement optimal, les **variables de conception** sont les suivantes:

$$\mathbf{x} = [\mathbf{H_{gr}}, \mathbf{D_{gr}}, \mathbf{N_t}, \mathbf{M_{gr}}]^T \tag{5.7}$$

Dans la mesure où les gerotor sont usinés sur base d'un catalogue de profils définissant son diamètre (six valeurs possibles) et son nombre de dents (deux valeurs possibles), les vecteurs $\mathbf{D_{gr}}$ et $\mathbf{N_t}$, tout comme $\mathbf{M_{gr}}$, présentent des variables discrètes. Par contre, la hauteur pouvant varier continûment, le vecteur $\mathbf{H_{gr}}$ a des composantes continues.

Les **contraintes** auxquelles sont soumises ces variables de conception proviennent des règles constitutives édictées par le client:

- la pompes d'alimentation (indicée $j = 1$) doit présenter un rotor intérieur à 6 dents:

$$N_t^1 = 6 \tag{5.8}$$

- la hauteur de chaque gerotor doit être comprise dans un intervalle spécifique (dont les bornes dépendent du type de pompe utilisée):

$$H_{gr}^j \in [h_{min}, h_{max}] \qquad j = 1, \ldots, n_p \tag{5.9}$$

avec:

$$h_{min} = \zeta_1 \; D_a \tag{5.10}$$

$$h_{max} = \zeta_2 \; D_a \tag{5.11}$$

où ζ_1 et ζ_2 sont deux paramètres fixés par le constructeur, et D_a désigne le diamètre au sommet de dents du rotor intérieur, fonction du nombre de dents et du diamètre de la pompe:

$$D_a = D_a\left(N_t^j, D_{gr}^j\right) \tag{5.12}$$

- chaque pompe, pour chaque cas de vol, doit fournir un débit d'huile spécifié par le client:

$$q_{min}^{i,j} \leq M_{gr}^{j} \; q_{gr}^{i,j} \qquad i = 1, \ldots, n_{fc} \qquad j = 1, \ldots, n_p \tag{5.13}$$

Le but final est de minimiser le coût, la masse et l'encombrement de l'empilage. Ces critères traduisent une optimisation multi-objectifs qui consiste, selon le souhait du partenaire industriel, à minimiser les trois fonctions suivantes:

- le nombre total de pompes dans l'empilage:

$$N_e = \sum_{j=1}^{n_p} M_{gr}^{j} \tag{5.14}$$

- la hauteur totale de l'empilage:

$$H_e = \sum_{j=1}^{n_p} M_{gr}^{j} \; H_{gr}^{j} \tag{5.15}$$

- le volume total de l'empilage (flasques non compris):

$$V_e = \sum_{j=1}^{n_p} M_{gr}^{j} \; H_{gr}^{j} \; \frac{\pi \left(D_{gr}^{j}\right)^2}{4} \tag{5.16}$$

Bien qu'ils puissent apparaître redondants, ces trois critères sont bel et bien antagonistes. En effet, si l'on diminue par exemple la multiplicité d'une fonction de pompe, la hauteur (et/ou le diamètre) des autres pompes devra nécessairement augmenter afin de satisfaire à nouveau les débits d'huile imposés par la spécification. On constate ainsi qu'une diminution du nombre total de pompes dans l'empilage s'accompagne généralement d'une augmentation de la hauteur (et/ou du volume) de cet empilage.

5.4 Résultats

GAGERO a été appliqué à deux cas réels proposés par Techspace Aero. Les valeurs suivantes ont été considérées pour les variables discrètes:

$$D_{gr}^{j} \in [31.5, 40.0, 50.0, 63.0, 80., 100.0] \tag{5.17}$$

$$N_t^{j} \in [4, 6] \tag{5.18}$$

$$M_{gr}^{j} \in [1, 2, 3] \tag{5.19}$$

Les deux paramètres technologiques qui sont utilisés pour spécifier les bornes de la hauteur des pompes (équations 5.10 et 5.11) ont été quant eux fixés à:

$$\zeta_1 = 0.12 \tag{5.20}$$

$$\zeta_2 = 1.25 \tag{5.21}$$

5.4.1 Cas 1: vitesse de rotation imposée

Dans ce premier problème, cinq fonctions de pompes et trois cas de vols sont pris en compte. La spécification du client est donnée au tableau 5.1.

Fonction de pompe	Cas de vol n°	N_a $[rpm]$	T_h $[^\circ C]$	P_{in} $[kPa]$	P_{out} $[kPa]$	q_s $[l/h]$
Alimentation (1)	1	6110	120	100	450	3215
	2	6110	120	50	350	2987
	3	6110	120	25	180	2344
Récupération AGB (2)	1	6110	160	100	450	3435
	2	6110	160	50	350	3136
	3	6110	160	25	180	2390
Récupération EAV (3)	1	6110	160	100	450	2300
	2	6110	160	50	350	2100
	3	6110	160	25	180	1600
Récupération EAR (4)	1	6110	160	100	450	2300
	2	6110	160	50	350	2100
	3	6110	160	25	180	1600
Récupération TGB (5)	1	6110	160	100	450	1140
	2	6100	160	50	350	1041
	3	6100	160	25	180	793

TAB. 5.1 – SPEC à vitesse imposée

Différentes combinaisons optimales de gerotor ont été identifiées par GAGERO. Les tableaux 5.2 et 5.3 présentent deux exemples, parmi les sept trouvés par le code, d'empilage optimal.

Fonction de pompe n°	H_{gr} [mm]	D_{gr} [mm]	N_t	M_{gr}
1	6.159	80.0	6	2
2	10.454	80.0	4	1
3	7.252	80.0	4	1
4	8.770	80.0	6	1
5	5.634	80.0	6	1

TAB. 5.2 – Géométrie de l'empilage optimal ♯1 (SPEC à vitesse imposée)

Le premier empilage comprend 6 pompes: 2 pompes d'alimentation et 4 pompes de récupération. Il présente une hauteur $H_e = 44.428\ mm$ et un volume $V_e = 223.332\ cm^3$. Le second empilage comporte 7 pompes: 2 pompes d'alimentation, 2 pompes de récupération pour l'enceinte AGB et une pompe pour chacune des autres fonctions de récupération (EAV, EAR et TGB). Cet empilage présente une hauteur légèrement plus élevée que la solution précédente: $H_e = 44.915\ mm$. Par contre, il est moins volumineux: $V_e = 211.199\ cm^3$.

Fonction de pompe n°	H_{gr} [mm]	D_{gr} [mm]	N_t	M_{gr}
1	5.951	80.0	6	2
2	5.670	80.0	4	2
3	7.019	80.0	4	1
4	7.014	80.0	4	1
5	7.640	63.0	6	1

TAB. 5.3 – Géométrie de l'empilage optimal ♮2 (SPEC à vitesse imposée)

A l'aide du tableau 5.4, on observe également le respect de la spécification du client. En effet, chacune des solutions proposées délivre bien, pour chaque fonction de pompe et chaque cas de vol, un débit d'huile (colonnes $q^{\natural 1}$ et $q^{\natural 2}$) supérieur ou égal à celui imposé par la SPEC (colonne q_s).

L'optimisation a été réalisée en utilisant une population de 500 individus dans l'algorithme génétique. Les solutions ont été obtenues après 20 itérations effectuées en 169 secondes.

Signalons enfin que la procédure empirique utilisée auparavant par Techspace Aero n'avait jamais fourni de résultats aussi probants en ce qui concerne ce cas. De plus, un système expert n'avait pu trouver une solution optimale en un temps raisonnable. Notons enfin que la résolution de ce cas, à l'aide du logiciel commercial BOSS Quattro, a conduit également à plusieurs empilages moins performants (hauteur et/ou volume plus élevé) que ceux obtenus par GAGERO et ce, après une dizaine d'heures de temps de calcul [3].

Fonction de pompe	Cas de vol n°	$q^{\natural 1}$ $[l/h]$	$q^{\natural 2}$ $[l/h]$	q_s $[l/h]$
Alimentation (1)	1	3371	3244	3215
	2	3282	3162	2987
	3	2452	2457	2344
Récupération AGB (2)	1	3585	3611	3435
	2	3483	3600	3136
	3	2675	3002	2390
Récupération EAV (3)	1	2394	2307	2300
	2	2365	2282	2100
	3	1919	1859	1600
Récupération EAR (4)	1	2361	2306	2300
	2	2295	2281	2100
	3	1765	1858	1600
Récupération TGB (5)	1	1408	1142	1140
	2	1397	1185	1041
	3	1138	1163	793

TAB. 5.4 – Débits d'huile fournis par les empilages ♮1 et ♮2 (SPEC à vitesse imposée)

5.4.2 Cas 2: vitesse de rotation libre

Dans ce second problème, deux pompes et trois cas de vol sont pris en compte. La spécification du client est donnée au tableau 5.5. Comme nous pouvons le constater, les vitesses de rotation de l'empilage ne sont plus imposées mais doivent être déterminées. Par conséquent, les variables de conception correspondent à:

$$\mathbf{x} = [\mathbf{H_{gr}}, \mathbf{D_{gr}}, \mathbf{N_t}, \mathbf{M_{gr}}, \mathbf{N_a}]^T \tag{5.22}$$

Fonction de pompe	Cas de vol n°	N_r [%]	T_h [$^\circ C$]	P_{in} [kPa]	P_{out} [kPa]	q_s [l/h]
Alimentation (1)	1	100	104.44	140.53	916.92	2928.09
	2	88.99	104.44	121.32	823.25	2794.20
	3	55.97	104.44	108.17	516.90	2178.83
Récupération RGB (2)	1	100	104.44	139.20	140.53	5863.13
	2	88.99	104.44	120.26	121.32	5594.80
	3	55.97	104.44	107.77	108.17	4361.88

TAB. 5.5 – SPEC à vitesse libre

Pour cette spécification à vitesse libre, GAGERO a déterminé une seule solution optimale définie aux tableaux 5.6 et 5.7. Cet empilage comprend 2 pompes, il présente une hauteur $H_e = 33.379\ mm$ et un volume $V_e = 262.167\ cm^3$.

Fonction de pompe n°	H_{gr} [mm]	D_{gr} [mm]	N_t	M_{gr}
1	13.461	100.0	6	1
2	19.918	100.0	4	1

TAB. 5.6 – Géométrie de l'empilage optimal ♮1 (SPEC à vitesse libre)

Cas de vol n°	N_a [rpm]
1	3990.826
2	3551.396
3	2233.545

TAB. 5.7 – Vitesses de rotation de l'empilage optimal (SPEC à vitesse libre)

A l'aide du tableau 5.8, on observe a nouveau le respect de la spécification du client. On remarque également que l'algorithme a réussi à adapter les vitesses de rotation "au plus juste" de façon à obtenir des débits d'huile très proches de ceux imposés par la SPEC. On constate ainsi que le troisième cas de vol est vraisemblablement le cas dimensionnant.

L'optimisation a été réalisée en utilisant une population de 300 individus dans l'algorithme génétique. La solution a été obtenue après 20 itérations effectuées en 49 secondes.

Fonction de pompe	Cas de vol n°	$q^{\sharp 1}$ $[l/h]$	q_s $[l/h]$
Alimentation (1)	1	3875.41	2928.09
	2	3443.91	2794.20
	3	2179.51	2178.83
Récupération RGB (2)	1	7813.98	5863.13
	2	6954.28	5594.80
	3	4375.02	4361.88

TAB. 5.8 – Débits d'huile fournis par l'empilage' $\sharp 1$ (SPEC à vitesse libre)

Notons enfin que la résolution de ce second problème, à l'aide du logiciel BOSS Quattro, a conduit à 3 solutions optimales obtenues après 6 heures de temps de calcul. Ces dernières correspondent à des empilages légèrement moins volumineux que celui proposé par GAGERO. Ces empilages présentent par contre des hauteurs et des vitesses de rotation nettement plus élevées [3].

Conclusions

L'examen de différentes stratégies d'optimisation a permis d'en définir les avantages et les points faibles, et de distinguer une approche particulièrement adaptée à la résolution de problèmes complexes d'optimisation: les algorithmes génétiques.

Cette métaheuristique appartient à la catégorie des recherches stochastiques d'ordre zéro. Elles est donc applicable à des fonctions objectifs discontinues mettant en présence des variables continues et/ou discrètes. De plus, basée sur l'évolution naturelle des populations, elles réalise une exploration guidée et intelligente de l'espace de conception. En outre, elle permet de prendre en compte, via la notion de dominance de Pareto, l'aspect multi-objectifs d'un problème d'optimisation.

Un code de calcul a été développé. Il est basé sur un codage réel des variables, et comprend les opérateurs génétiques classiques intervenant dans ce type de méthode. L'aspect multi-objectifs du problème a été traité via l'algorithme MOGA. Nous avons également proposé une approche originale qui permet de prendre en compte l'aspect contraint d'un problème d'optimisation, tout en utilisant la notion de dominance de Pareto présente dans l'algorithme MOGA.

Le code a été testé à partir de nombreux cas-tests mathématiques issus de la littérature. Il s'est montré apte à résoudre des problèmes d'optimisation mono et multi-objectifs, contraints ou non, en présence de variables continues et/ou discrètes. Il a en outre montré sa capacité à s'extraire d'optima locaux dans le cas d'une optimisation mono-objectif et a pu détecter des fronts de Pareto convexes, concaves ou discontinus dans le cas de problèmes multi-objectifs.

La méthode a enfin été appliquée avec succès au problème de dimensionnement optimal d'un empilage de pompes gerotor. A partir de deux cas réels fournis par Techspace Aero, l'algorithme a proposé rapidement plusieurs solutions optimales d'empilage qui satisfont les spécifications du client.

Enfin, l'aspect généraliste de la méthode employée, couplée à un structure modulaire du code de calcul, devrait permettre d'appliquer l'outil développé à d'autres problèmes de dimensionnement optimal. Dans le cadre des sciences appliquée, on peut citer le dimensionnement de systèmes mécaniques, automatiques, ou hydrauliques.

Bibliographie

[1] J. Baker. Adaptive Selection Methods for Genetic Algorithms. In J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithm and their Applications*, pages 101–111, Hillsdale, New Jersey, 1985. Lawrence Erlbaum Associates.

[2] T Bäck. Optimal Mutation Rates in Genetic Search. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 2–8, San Mateo, California, 1993. Morgan Kaufmann Publishers.

[3] P. Borremans. Développement d'une méthodologie de conception optimale d'un empilement de pompes de lubrification. Mémoire de fin d'études, Université de Liège, 2003.

[4] C. Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems*, 1(3):269–308, August 1999.

[5] I. Das and J. Dennis. A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. *Structural Optimization*, 14:63–69, 1997.

[6] Ingenet Test-Cases Database. `http://www-sop.inria.fr/sinus/ingenet/`.

[7] M. Dorigo. *Optimization, Learning and Natural Algorithms*. Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[8] E. Easom. A Survey of Global Optimization Techniques. Master Engineering Thesis, University of Louisville, 1990.

[9] H. Eschenauer, J. Koski, and A. Osyczka. *Multicriteria Design Optimization: Procedures and Applications*. Springer-Verlag, Berlin, 1990.

[10] L. Eshelman and J. Schaffer. *Real-Coded Genetic Algorithms and Interval Schemata*. Foundations of Genetic Algorithms. Morgan Kaufmann Publishers, Inc., 1993.

[11] C. Fleury. *Optimisation des structures*. Notes de cours. Université de Liège, 1995.

[12] C. Fleury and V. Braibant. Structural Optimization: A New Dual Method Using Mixed Variables. Technical Report SA-115, Université de Liège, 1984.

[13] L. Fogel, A. Owens, and M. Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley and Sons, New-York City, 1966.

[14] C. Fonseca and P. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, University of Illinois, 1993. Morgan Kauffman Publishers.

[15] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[16] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company Inc., 1989.

[17] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.

[18] J. Horn and N. Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithms. Technical report, University of Illinois, USA, 1993.

[19] J. Joines and C. Houck. On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GAs. In *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pages 579–584. IEEE Press, 1994.

[20] V. Kelner and O. Léonard. Application of Genetic Algorithms to Lubrication Pump Stacking Design. *Journal of Computational and Applied Mathematics*, In Press.

[21] S. Kirkpatrick, C. Gelatt, and P. Vechhi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[22] J. Koski and R. Silvennoinen. Norm Methods and Partial Weighting in Multicriterion Optimization of Structures. *International Journal of Methods in Engineering*, 24:1101–1121, 1987.

[23] A. Kreis. *Optimisation multiobjectifs des sytèmes dynamiques. Application à la suspension de groupes motopropulseurs de véhicules automobiles en phase d'avant-projet.* Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambresis, 2001.

[24] K. KrishnaKumar. Building Blocks of Evolutionary Algorithms. In *Genetic Algorithms for Optimisation in Aeronautics and Turbomachinery*, von Karman Institute, 2000. Lecture Series 2000-07.

[25] H. Muhlenbein and D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm: Continuous Parameter Optimization, pages 25-49. *Evolutionary Computation*, 1(1):25–49, 1993.

[26] A. Osyczka. *Design Optimization*, chapter Multicriteria Optimization for Engineering Design, pages 193–227. J. Gero, Academic Press edition, 1985.

[27] V. Pareto. *Cours d'économie politique*. F. Rouge, Lausanne, 1896.

[28] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.

[29] J. Richardson, M. Palmer, G. Liepins, and M. Hilliard. Some Guidelines for Genetic Algorithms with Penalty Functions. In J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197, George Mason University, 1989. Morgan Kaufmann Publishers.

[30] J. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In Lawrence Erlbaum, editor, *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100, 1985.

[31] H. Schwefel. *Numerical Optimization of Computers Models*. Wiley and Sons, Chichester, 1981.

[32] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Technical report, Indian Institute of Technology (Department of Mechanical Engineering), Kanput, India, 1993.

[33] W. Stadler. Caveats and Boons of Multicriteria Optimization. *Microcomputers in Civil Engineering*, 10(4):291–299, 1995.

[34] K. Svanberg. The Method of Moving Asymptotes: A New Method for Structural Optimization. *International Journal of Numerical Methods in Engineering*, 24:359–373, 1987.

[35] K. Svanberg. A Globally Convergent Version of MMA without Linesearch. In *Proceedings of the First Word Congress on Structural and Multidisciplinary Optimization*, pages 9–16, Oxford, 1995.

[36] D. Van Veldhuizen and G. Lamont. *Multiobjective Evolutionary Algorithms Research: A History and Analysis.* Department of Electrical and Computer Engineering - Graduate School of Engineering - Air Force Institute of Technology, Wright-Patterson AFB, OH 45433-7765, 1998.

[37] G. Vanderplaats. *Numerical Optimization Techniques for Engineering Design with Applications.* Mc Graw-Hill, 1984.

[38] W. Zhang and C. Fleury. A Generalized Method of Moving Asymptotes. In *Proceedings of the WCSMO Conference*, Germany, 1995.

[39] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich, 1999.

[40] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):174–195, 2000.