UNIVERSITÉ
D'ANGERS

Année : 2003
N° d'ordre : 593

APPROCHES HYBRIDES POUR LES PROBLÈMES MULTIOBJECTIFS

THÈSE DE DOCTORAT

Spécialité : Informatique

ÉCOLE DOCTORALE D'ANGERS

Présentée et soutenue publiquement

Le 24 Novembre 2003

À Angers

Par **Vincent BARICHARD**

Devant le jury ci-dessous :

Président :	Jacques TEGHEM,	Professeur à la Faculté Polytechnique de Mons
Les rapporteurs :	Marc SCHOENAUER,	Directeur de Recherche à l'INRIA
	Patrick SIARRY,	Professeur à l'Université de Paris XII Val-de-Marne
Les examinateurs :	Jean-Jacques CHABRIER,	Professeur à l'Université de Bourgogne
	Éric JACQUET-LAGRÈZE,	Président Directeur Général d'Eurodécision
Directeur de thèse :	Jin-Kao HAO,	Professeur à l'Université d'Angers

Laboratoire d'Etude et de Recherche en Informatique d'Angers
2, Bd Lavoisier, 49045 Angers Cedex 01

Remerciements

Je tiens à remercier en tout premier lieu Jin-Kao Hao, mon directeur de thèse, pour m'avoir accueilli, soutenu et encadré constamment pendant les trois années de cette thèse.

Je remercie Marc Schoenauer et Patrick Siarry, tous deux rapporteurs de cette thèse, pour avoir lu de manière approfondie mon manuscrit. Leurs précieux conseils m'ont permis d'en améliorer le fond et la forme.

Je remercie également les examinateurs Jean-Jacques Chabrier, Éric Jacquet-Lagrèze et Jacques Teghem pour l'intérêt qu'ils ont porté à mon travail.

Je remercie tous les membres du projet `METEOR` et en particulier Fabrice Chauvet, Éric Jacquet-Lagrèze (à nouveau ☺) et Laurent Pajou. Les différents échanges que nous avons pu avoir m'ont permis de faire évoluer mes idées et de me donner une autre vision de la recherche.

Je remercie chaleureusement tous les membres de l'équipe Métaheuristiques et Optimisation Combinatoire, et en particulier Frédéric Saubion et Jean-Michel Richer pour leur disponibilité, leur écoute et leurs conseils toujours judicieux. Je remercie également tous les membres du département informatique de l'université d'Angers : Tassadit Amghar, Florence Bannay, Catherine Boisseau, Annick Corolleur, Frantz De Germain, Béatrice Duval, Laurent Garcia, David Genest, Éric Girardeau, Jean-Philippe Hamiez, Gilles Hunault, Claire Lefèvre, Bernard Levrat, Stéphane Loiseau, Pascal Nicolas, Daniel Pacholczyk, Igor Stéphan, Stéphane Vincendeau et tous les autres, avec qui j'ai discuté ou pris une pause café et qui rendent la vie au laboratoire agréable.

Je remercie Xavier Baguenard et Massa Dao du LISA. Notre collaboration m'a permis de faire de grands pas dans l'exploitation de mes résultats.

Je remercie tout particulièrement Luc Jaulin qui m'a guidé et encouragé à un moment où j'en avais besoin. Sa bonne humeur, ses conseils et sa disponibilité m'ont permis d'avancer dans le bon sens.

Je n'oublie pas mes compagnons thésards : Olivier Cantin, Hervé Deleau, Vincent Derrien, Styve Jaumotte, Frédéric Lardeux, Antoine Robin et les autres. Les moments de détente passés ensemble sont une véritable bouffée d'oxygène lorsque le stress est omniprésent.

Je dois un grand merci à Narendra Jussien qui m'a fait faire mes premiers pas dans la recherche et qui a continué de me soutenir tout au long de mon parcours. Son amitié restera une des meilleures découvertes de cette thèse.

Je ne pourrais oublier mes amis qui m'ont accompagné durant cette période. Je leur adresse toute ma reconnaissance pour leur amitié et les sorties souvent mémorables : Mériéma et Michaël Aupetit, Nathalie Bricard, Denis Carré, Valérie et Thomas Dionisi, Denis Flamant, Nordine Fourour, Sibylle Héron, Anne et Benoît Huchedé, Nicolas Gourichon, Christelle et Narendra Jussien, Hélène et Brice Pipino, Sophie Renaud, Barbara et Sébastien Trottier, Nicolas Vaillant.

Ces remerciements ne seraient pas complets sans y avoir associé ma famille. Je remercie tout d'abord mes parents pour leur soutien moral, leur présence et leur écoute. Ils ont su

m'épauler et me faire confiance pendant toutes ces années. Je remercie également ma belle-famille pour m'avoir accueilli chaleureusement tant de fois.

Enfin, je remercie du fond du cœur ma Cécile. Son soutien sans failles, ses bons petits plats, son réconfort quotidien ont eu raison de toutes mes angoisses.

À Cécile l'inspiration de ma vie.

Table des matières

Partie IV Extensions et ouvertures

Introduction générale

Présentation

Résoudre un problème d'optimisation consiste à trouver la ou les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur. Pour déterminer si une solution est meilleure qu'une autre, il est nécessaire que le problème introduise un critère de comparaison. Ainsi, la meilleure solution, appelée aussi solution optimale, est la solution ayant obtenu la meilleure valuation au regard du critère défini. Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes appliqués : le traitement d'images, la conception de systèmes, la conception d'emplois du temps, ... La majorité de ces problèmes sont qualifiés de *difficiles*, car leur résolution nécessite l'utilisation d'algorithmes évolués, et il n'est en général pas possible de fournir dans tous les cas une solution optimale dans un temps *raisonnable*. Lorsqu'un seul critère est donné, par exemple un critère de minimisation de coût, la solution optimale est clairement définie, c'est celle qui a le coût minimal. Mais dans de nombreuses situations, un seul critère peut être insuffisant. En effet, la plupart des applications traitées intègrent plusieurs critères simultanés, souvent contradictoires. Intégrer des critères contradictoires pose un réel problème. Considérons les actions suivantes :

- Louer un appartement bien situé et d'un prix raisonnable.
- Établir un planning pour les vacances satisfaisant toute la famille.
- Choisir entre plusieurs itinéraires.
- Acheter une voiture.

La plupart des critères liés à ces tâches sont contradictoires. La solution idéale n'existe pas, et il faut donc trouver un compromis. En effet, en considérant deux critères contradictoires a et b, améliorer a détériore forcément b et inversement. Le concept de solution optimale devient alors plus difficile à définir. Dans ce cas, la solution optimale cherchée n'est plus un point unique, mais un ensemble de compromis. Résoudre un problème comprenant plusieurs critères, appelé communément problème multiobjectif, consiste donc à calculer le meilleur ensemble de solutions compromis : *le front Pareto*. Dans cette thèse, nous traiterons principalement de la résolution de problèmes multiobjectifs.

Approches de résolution

Qu'ils comportent un seul ou plusieurs objectifs, les problèmes d'optimisation sont en général difficiles à résoudre. De plus, le temps de calcul nécessaire à leur résolution peut devenir si important que l'algorithme développé devient inutilisable en pratique. Il n'existe pas d'algorithme générique capable de résoudre toutes les instances de tous les problèmes efficacement. Un grand nombre de méthodes ont été développées pour tenter

d'apporter une réponse satisfaisante à ces problèmes. Parmi celles-ci, nous distinguons les méthodes dédiées à un problème et les méthodes plus génériques pouvant s'appliquer à un ensemble de problèmes. Les méthodes dédiées utilisent des heuristiques propres au problème permettant une meilleure adaptation de l'algorithme à ce problème particulier. Dans cette thèse, nous traiterons des méthodes génériques pouvant s'appliquer à plusieurs problèmes avec un minimum de réécriture de l'algorithme. Parmi ces méthodes génériques, nous distinguons deux grandes classes de méthodes : *les méthodes exactes* et *les méthodes approchées*.

Les méthodes exactes examinent, souvent de manière implicite, la totalité de l'espace de recherche. Ainsi, elles ont l'avantage de produire une solution optimale lorsqu'aucune contrainte de temps n'est donnée. Néanmoins, le temps de calcul nécessaire pour atteindre une solution optimale peut devenir vite prohibitif, ce malgré les diverses techniques et heuristiques qui ont été développées pour accélérer l'énumération des solutions.

Dans de telles situations (et dans beaucoup d'autres), les méthodes approchées constituent une alternative indispensable et complémentaire. Le but d'une telle méthode n'est plus de fournir une solution optimale au problème donné. Elle cherche avant tout à produire une solution sous-optimale de meilleure qualité possible avec un temps de calcul raisonnable. En général, une méthode approchée examine seulement une partie de l'espace de recherche. Le choix des solutions examinées est souvent dicté par des heuristiques.

À partir de deux méthodes de résolution différentes, il est possible de concevoir des méthodes hybrides dans le but de fournir des résultats supérieurs aux deux méthodes qui les composent. Dans cette thèse, nous étudions la combinaison de méthodes pour la résolution des problèmes multiobjectifs. Nous nous penchons plus particulièrement sur des méthodes combinant des métaheuristiques et sur des méthodes combinant une approche exacte et une approche approchée.

Cadre de travail

À travers cette thèse, nous cherchons à apporter des éléments de réponses sur plusieurs problèmes et questions que nous nous sommes posés. Tout d'abord, nous nous sommes fixés comme cadre de travail les problèmes multiobjectifs. Par rapport à l'optimisation mono-objectif, les recherches sur les problèmes multiobjectifs sont récentes (une trentaine d'années) et naturellement, ces problèmes sont plus méconnus que les problèmes à un seul objectif. Les premiers travaux menés sur l'application des méthodes approchées aux problèmes multiobjectifs sont fondés sur les principaux algorithmes et résultats proposés dans le cadre de la résolution des problèmes d'optimisation classiques (ne comportant qu'un seul objectif). Cette réutilisation de la connaissance a permis d'apporter rapidement des réponses pour résoudre ces nouveaux problèmes d'optimisation, en adaptant les algorithmes de résolution déjà existants. Mais de nouvelles difficultés propres aux problèmes d'optimisation multiobjectifs sont apparues faisant sentir la nécessité d'approfondir nos connaissances sur ces problèmes. Notre premier objectif consiste à étudier certains aspects influant fortement sur la résolution des problèmes multiobjectifs, notamment l'importance de la diversification. Nous espérons qu'une fois ces aspects identifiés et étudiés,

nous pourrons construire une méthode hybride (une métaheuristique) combinant plusieurs approches, chacune répondant à un des aspects étudiés.

Au début de cette thèse, nous pressentions que les méthodes approchées seules ne pourraient répondre à toutes les exigences posées par les problèmes multiobjectifs. En effet, calculer une solution approchée est une chose, calculer une approximation de tout un ensemble de solutions compromis en est une autre. Mesurer la qualité de cet ensemble, et sa distance à l'ensemble optimal, sont autant d'interrogations auxquelles il nous semblait difficile de répondre avec des méthodes approchées. Les méthodes exactes quant à elles ne sont pas assez efficaces pour répondre aux besoins de l'utilisateur. Dans un deuxième temps nous avons donc souhaité étudier les différents *ponts* existant entre méthodes exactes et approchées, et essayer de conserver au mieux les avantages de chacune des approches.

Principales contributions

Les travaux effectués ont donné lieu à plusieurs contributions de différents types. La première catégorie regroupe les contributions algorithmiques. Elles se retrouvent sous la forme d'algorithmes développés dans le cadre des études menées. Ainsi, pour le problème du sac à dos multidimensionnel multiobjectif, nous avons développé un algorithme Tabou original, appelé `TS+HW` (*Tabu Search + Hamming Watch*), permettant de mettre en évidence l'importance de la gestion de la diversité lors de l'optimisation d'un problème multiobjectif. Pour ce même problème, nous avons développé ensuite un algorithme hybride, GTS^{MOKP} (*Genetic Tabu Search*), combinant une méthode de recherche Tabou et une méthode évolutionnaire, chaque composant ayant une tâche déterminée : *intensifier* ou *diversifier*.

Une grande partie du travail de cette thèse concerne la conception, le développement et la validation de l'algorithme `PICPA` (*Population and Interval Constraint Propagation*). `PICPA` combine des méthodes exactes et approchées. En effet, `PICPA` intègre des mécanismes de propagation de contraintes et d'analyse par intervalles avec des concepts évolutionnaires. `PICPA` est un algorithme déterministe calculant des solutions approchées tout en déterminant des bornes du front Pareto. C'est, à notre connaissance, le premier algorithme effectuant ces deux actions simultanément, de manière complémentaire. Nous avons aussi développé `PICPA-II`, une amélioration de `PICPA`, qui intègre une sélection Pareto améliorée ainsi qu'une méthode de recherche locale pour accélérer la recherche d'une solution particulière. Cette méthode de recherche locale intègre un processus inspiré des stratégies d'évolution combiné à des mécanismes de propagation de contraintes. Bien qu'intégrant des mécanismes exacts, `PICPA-II` rivalise en temps de calcul avec les métaheuristiques utilisées habituellement. En contrepartie il perd l'aspect déterministe présent dans `PICPA`.

La deuxième catégorie de contributions regroupe les aspects plus fondamentaux de nos travaux. Nous avons tout d'abord étudié certains aspects de la résolution des problèmes multiobjectifs et ainsi mis en évidence l'importance des phases de diversification. Nous avons étendu une procédure de mesure de la diversité pour la rendre incrémentale. Nous avons aussi défini une extension de la relation classique de dominance, permettant de com-

parer un point à un ensemble. Cette nouvelle relation est mise en application dans nos algorithmes *PICPA* et *PICPA-II*. De plus, nous avons proposé de nouvelles définitions, énoncé et démontré des propriétés permettant d'étendre l'utilisation des outils de résolution de l'optimisation multiobjectif à des problèmes d'approximation d'ensembles. Dans ce but, nous avons notamment introduit les concepts d'axe-convexité et de frontière Pareto.

Organisation du document

L'organisation de cette thèse est la suivante : la première partie dresse un état de l'art non exhaustif des domaines de l'optimisation multiobjectif et des métaheuristiques. Au chapitre 1, nous présentons l'optimisation multiobjectif. Nous introduisons des concepts fondamentaux tels que la dominance, la surface de compromis. Nous décrivons aussi les principales approches non Pareto de résolution pour ces problèmes. Au chapitre 2, nous présentons quelques approches Pareto, appartenant à la classe des métaheuristiques, dédiées aux problèmes d'optimisation multiobjectifs. Au chapitre 3, nous présentons quelques mesures permettant d'évaluer un aspect de la surface de compromis produite par un algorithme.

La deuxième partie de cette thèse concerne les travaux menés sur les phases de diversification des métaheuristiques dans le cadre de problèmes discrets, et plus particulièrement sur le problème du sac à dos multidimensionnel multiobjectif. Au chapitre 4, nous démontrons expérimentalement qu'une des clefs pour réaliser un algorithme performant, est liée à l'incorporation d'un mécanisme *intelligent* de diversification à la méthode. Pour illustrer nos propos, nous présentons trois algorithmes de recherche Tabou pour le sac à dos multidimensionnel multiobjectif. Au chapitre 5, nous nous intéressons à l'approche hybride et présentons GTS^{MOKP}, un algorithme génétique combiné à une recherche Tabou pour le sac à dos multidimensionnel multiobjectif.

La troisième partie traite des problèmes multiobjectifs continus. Nous développons dans cette partie un schéma hybride combinant méthodes exactes et approchées pour leur résolution. Au chapitre 6, nous présentons *PICPA*, un nouvel algorithme pour traiter les problèmes multiobjectifs continus sous contraintes. Cet algorithme combine des techniques de propagation de contraintes et des concepts évolutionnaires. *PICPA* est capable de calculer des bornes du front Pareto optimal tout en produisant des solutions approchées très précises. Au chapitre 7, nous présentons *PICPA-II*, une amélioration de notre précédent algorithme hybride. *PICPA-II* intègre une nouvelle heuristique de recherche mélangeant *recherche locale* et *coupes dichotomiques*.

La quatrième partie concerne des aspects plus fondamentaux, déduits des travaux effectués dans les parties précédentes. Ainsi, au chapitre 8, nous présentons de nouvelles définitions établissant le lien entre les problèmes multiobjectifs et l'approximation *d'enveloppes axe-convexes* d'ensembles. Nous énonçons et démontrons des propriétés permettant d'effectuer cette jonction, et montrons une application de ce type de problèmes.

Enfin, nous finissons par une conclusion générale reprenant les différentes contributions apportées dans cette thèse, analysant les limites de ces apports et donnant quelques perspectives de recherche.

Première partie

État de l'art de l'optimisation multiobjectif

Chapitre 1

L'optimisation multiobjectif

Dans ce chapitre, nous présentons l'optimisation multiobjectif qui sera le cadre de travail de cette thèse. Nous introduisons les concepts fondamentaux tels que la dominance, la surface de compromis. Nous décrivons aussi les principales approches non Pareto de résolution pour ces problèmes.

Sommaire

1.1 Problèmes d'optimisation mono-objectifs

Résoudre un problème d'optimisation consiste à trouver une solution qui minimise ou maximise un critère particulier. Dans la plupart des cas, l'optimum découvert n'est pas unique. Ainsi il existe un ensemble de solutions minimisant ou maximisant le critère considéré. Nous pouvons décrire formellement un problème d'optimisation comme suit :

Considérons un problème de minimisation :

Soit $f : \mathbb{R}^n \rightarrow \mathbb{R}$,
et soit $\mathcal{X}$ un ensemble fermé de $\mathbb{R}^n$
alors l'ensemble des minimiseurs est : $\hat{\mathcal{X}} = \{\overrightarrow{x} \in \mathcal{X} | \, \forall \overrightarrow{x'} \in \mathcal{X}, \; f(\overrightarrow{x'}) \geq f(\overrightarrow{x})\}$
et le minimum du problème est : $f(\hat{\mathcal{X}})$

L'ensemble $\mathcal{X}$ représente l'ensemble des solutions potentielles du problème. Cet ensemble est déterminé à l'aide de contraintes (souvent analytiques) données dans l'énoncé du problème. Le formalisme précédent ne faisant pas explicitement intervenir les contraintes du problème, un problème d'optimisation mono-objectif est plus souvent donné sous la forme suivante :

Minimiser	$f(\overrightarrow{x})$	(fonction à optimiser)
tel que	$\overrightarrow{g}(\overrightarrow{x}) \leq 0$	(q contraintes à satisfaire)
avec	$\overrightarrow{x} \in \mathbb{R}^n, \; \overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^q$	

Ces deux écritures sont équivalentes, la deuxième étant le plus souvent rencontrée dans les domaines de *l'Intelligence Artificielle* et de *l'Optimisation*. Il est clair que pour minimiser (resp. maximiser) une fonction d'un problème, il faut déjà être capable de déterminer l'ensemble des solutions potentielles. On distingue ainsi deux notions : la notion *d'espace de recherche* représentant l'ensemble des valeurs pouvant être prises par les variables, et la notion *d'espace réalisable* représentant l'ensemble des valeurs des variables satisfaisant les contraintes. Dans le cadre de cette thèse, nous traiterons de problèmes appartenant à la classe de problèmes *NP-difficiles*. Nous revenons sur cette notion au chapitre 2.

Sur la figure 1.1, $\underline{x_i}$ (resp. $\overline{x_i}$) représente la borne inférieure (resp. supérieure) donnée pour x_i, $\mathcal{C}$ représente l'espace de recherche égal au produit cartésien des domaines des variables, et $\mathcal{X}$ l'espace réalisable délimité par les contraintes. Cette figure, illustre en dimension 2 le fait que $\mathcal{X} = \{\overrightarrow{x} \mid \overrightarrow{g}(\overrightarrow{x}) \leq 0, \overrightarrow{x} \in \mathcal{C}\}$ l'espace réalisable (délimité par des contraintes) est un sous-ensemble de $\mathcal{C}$ (ici $\mathcal{C} = \mathbb{R}^2$) l'espace de recherche.

Sauf mention contraire explicite, tous les énoncés et définitions seront donnés dans le cadre de problèmes de minimisation. En effet un problème de maximisation peut être aisément transformé en problème de minimisation en considérant l'équivalence suivante :

$$\text{maximiser } f(\overrightarrow{x}) \Longleftrightarrow \text{ minimiser } -f(\overrightarrow{x})$$

Pour la suite de cette thèse, les vecteurs pourront être notés de manière indifférente $\overrightarrow{x}$ ou $\mathbf{x}$.

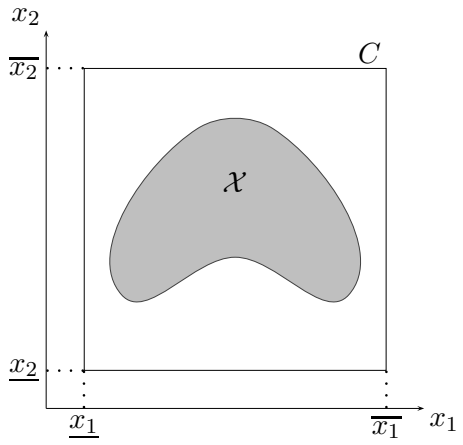


FIG. 1.1 – *Espace de recherche et espace réalisable*

1.2 Vocabulaire et définitions

Nous donnons ici quelques définitions des principaux termes fréquemment utilisés dans cette thèse :

Définition 1 Une boîte ou pavé : *C'est le nom donné au produit cartésien d'intervalles fermés. L'ensemble $\mathcal{C}$ de la figure 1.1 est le pavé résultant du produit cartésien des intervalles $[x_1]$ et $[x_2]$.*

Définition 2 Vecteur de décision : *C'est le nom donné au vecteur $\overrightarrow{x}$. Il correspond à l'ensemble des variables du problème.*

Définition 3 Fonction objectif : *C'est le nom donné à la fonction f (appelé aussi fonction de coût ou critère d'optimisation).*

Définition 4 Minimum global : *Un point $\overrightarrow{x} \in \mathcal{X}$ est un minimum global du problème si et seulement si : $\forall \overrightarrow{x'} \in \mathcal{X},\ f(\overrightarrow{x}) \leq f(\overrightarrow{x'})$ (il appartient alors à l'ensemble des minimiseurs).*

Définition 5 Minimum local : *Un point $\overrightarrow{x} \in \mathcal{X}$ est un minimum local du problème si et seulement si : $\forall \overrightarrow{x'} \in \mathcal{V}(\mathcal{X}),\ f(\overrightarrow{x}) \leq f(\overrightarrow{x'})$, où $\mathcal{V}(\mathcal{X})$ définit un "voisinage" de $\overrightarrow{x}$.*

1.3 Problèmes d'optimisation multiobjectifs

La plupart des problèmes d'optimisation réels sont décrits à l'aide de plusieurs objectifs ou critères souvent contradictoires devant être optimisés simultanément. Alors que, pour les problèmes n'incluant qu'un seul objectif, l'optimum cherché est clairement défini, celui-ci reste à formaliser pour les problèmes d'optimisation multiobjectifs. En effet, pour un problème à deux objectifs contradictoires, la solution optimale cherchée est un ensemble de points correspondant aux meilleurs compromis possibles pour résoudre notre problème.

Prenons le cas d'une personne souhaitant acheter une voiture d'occasion. La voiture idéale est celle qui est peu chère avec peu de kilomètres, mais cette voiture idyllique n'existe pas (sinon je l'aurais achetée). Notre acheteur va donc devoir identifier les meilleurs compromis possibles correspondant à son budget (voir figure 1.2).

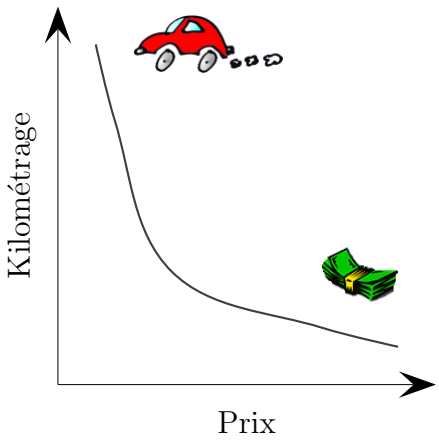


FIG. 1.2 – *Relation entre kilométrage et prix.*

Les problèmes d'optimisation multiobjectifs sont une généralisation à n fonctions objectif des problèmes d'optimisation classiques (cf. Section 1.1). Ils sont définis formellement comme suit :

Considérons un problème de minimisation :
Soit $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$,
et soit $\mathcal{X}$ un ensemble fermé de $\mathbb{R}^n$
alors l'ensemble des minimiseurs est :

$$\hat{\mathcal{X}} = \left\{ \overrightarrow{x} \in \mathcal{X} | \, \forall \overrightarrow{x'} \in \mathcal{X}, \left(\exists i \in [1, \cdots, m] \; f_i(\overrightarrow{x}) < f_i(\overrightarrow{x'}) \right) \right. \\ \left. \vee \left(\forall i \in [1, \cdots, m] \; f_i(\overrightarrow{x}) = f_i(\overrightarrow{x'}) \right) \right\}$$

et le minimum du problème est : $f(\hat{\mathcal{X}})$

D'après cette définition, il est clair que l'optimum n'est plus une simple valeur comme pour les problèmes à un objectif, mais un ensemble de points, appelé l'ensemble des meilleurs compromis ou le *front Pareto*. Dans la suite de cette thèse, nous adopterons la formulation suivante, équivalente à la précédente, mais plus souvent utilisée dans les travaux actuels :

Minimiser	$\overrightarrow{f}(\overrightarrow{x})$	(m fonctions à optimiser)
tel que	$\overrightarrow{g}(\overrightarrow{x}) \leq 0$	(q contraintes à satisfaire)
avec	$\overrightarrow{x} \in \mathbb{R}^n, \; \overrightarrow{f}(\overrightarrow{x}) \in \mathbb{R}^m, \; \overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^q$	

1.4 Relations d'ordre et de dominance

Comme la solution optimale est une multitude de points de $\mathbb{R}^m$, il est vital pour identifier ces meilleurs compromis de définir une relation d'ordre entre ces éléments. Dans le cas des problèmes d'optimisation multiobjectifs, ces relations d'ordre sont appelées *relations de dominance*. Plusieurs relations de dominance ont déjà été présentées : la *a-dominance* [Othmani, 1998], la *dominance au sens de Geoffrion* [Ehrgott, 2000], la *cône-dominance* [Collette and Siarry, 2002], ...

Mais la plus célèbre et la plus utilisée est la *dominance au sens de Pareto*. C'est cette relation de dominance que nous allons définir et utiliser dans cette thèse. De manière à définir clairement et formellement cette notion, les relations $=$, $\leq$ et $<$ usuelles sont étendues aux vecteurs.

Définition 6 *Soient* $\mathbf{u}$ *et* $\mathbf{v}$*, deux vecteurs de même dimension,*

$$\begin{array}{lll} \mathbf{u} = \mathbf{v} & ssi & \forall i \in \{1, 2, \cdots, m\},\ u_i = v_i \\ \mathbf{u} \leq \mathbf{v} & ssi & \forall i \in \{1, 2, \cdots, m\},\ u_i \leq v_i \\ \mathbf{u} < \mathbf{v} & ssi & \mathbf{u} \leq \mathbf{v} \wedge \mathbf{u} \neq \mathbf{v} \end{array}$$

Les relations $\geq$ *et* $>$ *sont définies de manière analogue.*

Les relations définies précédemment ne couvrent pas tous les cas possibles. En effet, il est impossible de classer les points $\mathbf{a} = (1, 2)$ et $\mathbf{b} = (2, 1)$ à l'aide d'une de ces relations. Contrairement aux problèmes à un seul objectif où les relations usuelles $<$, $\leq$, ... suffisent pour comparer les points, elles sont insuffisantes pour comparer des points issus de problèmes multiobjectifs. Nous définissons donc maintenant la relation de dominance au sens de Pareto permettant de prendre en compte tous les cas de figures rencontrés lors de la comparaison de deux points (ici des vecteurs).

Définition 7 La dominance au sens de Pareto : *Considérons un problème de minimisation. Soient* $\mathbf{u}$ *et* $\mathbf{v}$ *deux vecteurs de décision,*

$$\begin{array}{rcl} \mathbf{u} \prec \mathbf{v}\ (\mathbf{u}\ \textit{domine}\ \mathbf{v}) & ssi & \mathbf{f}(\mathbf{u}) < \mathbf{f}(\mathbf{v}) \\ \mathbf{u} \preceq \mathbf{v}\ (\mathbf{u}\ \textit{domine faiblement}\ \mathbf{v}) & ssi & \mathbf{f}(\mathbf{u}) \leq \mathbf{f}(\mathbf{v}) \\ \mathbf{u} \sim \mathbf{v}\ (\mathbf{u}\ \textit{est incomparable (ou non-dominé) avec}\ \mathbf{v}) & ssi & \mathbf{f}(\mathbf{u}) \not\leq \mathbf{f}(\mathbf{v}) \wedge \mathbf{f}(\mathbf{v}) \not\leq \mathbf{f}(\mathbf{u}) \end{array}$$

Pour un problème de maximisation, ces relations sont définies de manière symétrique.

Définition 8 *Une solution* $\mathbf{x} \in \mathcal{X}_f$ *est dite non dominée par rapport à un ensemble* $\mathcal{X}_a \subseteq \mathcal{X}_f$ *si et seulement si :*

$$\nexists \mathbf{x}_a \in \mathcal{X}_a,\ \mathbf{x}_a \prec \mathbf{x}$$

Illustrons maintenant cette relation par un exemple en dimension 2 (cf. figure 1.3). Sur cette figure, $\mathcal{F}$ (l'espace réalisable dans l'espace des objectifs) est l'image de $\mathcal{X}$. Ainsi, chaque point $\mathbf{y^i}$ est l'image de $\mathbf{x^i}$ par f : $\mathbf{y^i} = f(\mathbf{x^i})$. Prenons le point $\mathbf{y^1}$ comme point de référence. Nous pouvons distinguer trois zones :

– La zone de préférence est la zone contenant les points dominés par $\mathbf{x^1}$,

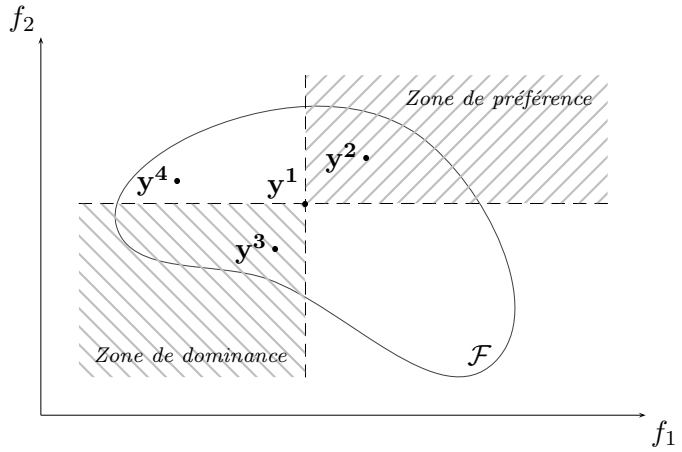


FIG. 1.3 – *Exemple de dominance.*

– La zone de dominance est la zone contenant les points dominant $\mathbf{x^1}$,

– La zone d'incompatibilité contient les points incomparables avec $\mathbf{x^1}$.

Ainsi, il est clair que $\mathbf{x^2}$ est dominé par $\mathbf{x^1}$ ($\mathbf{x^1}$ est préféré à $\mathbf{x^2}$), que $\mathbf{x^3}$ domine $\mathbf{x^1}$ ($\mathbf{x^3}$ est préféré à $\mathbf{x^1}$), et que $\mathbf{x^4}$ est non dominé (incomparable) avec $\mathbf{x^1}$.

Forts de ce nouvel outil, nous pouvons maintenant définir l'optimalité dans le cas de problèmes multiobjectifs. Nous pouvons définir l'optimalité globale au sens de Pareto (c.à.d utilisant la dominance au sens de Pareto) :

Définition 9 L'optimalité globale au sens de Pareto : *Un vecteur de décision* $\mathbf{x} \in \mathcal{X}$ *est dit Pareto globalement optimal si et seulement si :* $\nexists \mathbf{y} \in \mathcal{X},\ \mathbf{y} \prec \mathbf{x}$. *Dans ce cas,* $\mathbf{f}(\mathbf{x}) \in \mathcal{F}$ *est appelé :* **solution efficace**.

Nous pouvons maintenant définir le concept d'optimalité locale au sens de Pareto :

Définition 10 L'optimalité locale au sens de Pareto : *Un vecteur de décision* $\mathbf{x} \in \mathcal{X}$ *est dit Pareto optimal localement si et seulement si, pour un* $\delta > 0$ *fixé :* $\nexists y \in \mathcal{X},\ \mathbf{f}(\mathbf{y}) \in B(\mathbf{f}(\mathbf{x}), \delta)$ *et* $y \prec x$, *où* $B(\mathbf{f}(\mathbf{x}), \delta)$ *représente une boule de centre* $\mathbf{f}(\mathbf{x})$ *et de rayon* δ.

La figure 1.4 donne un exemple en dimension 2 d'optimalité locale. Le point $\mathbf{f}(\mathbf{x})$ est localement optimal, car il n'y a pas de point compris dans la boule B le dominant.

1.5 Front Pareto et surface de compromis

Nous rappelons que la solution que nous cherchons à un problème d'optimisation multiobjectif n'est pas un point unique mais un ensemble de points que nous avons appelé l'ensemble des meilleurs compromis à la Section 1.3.

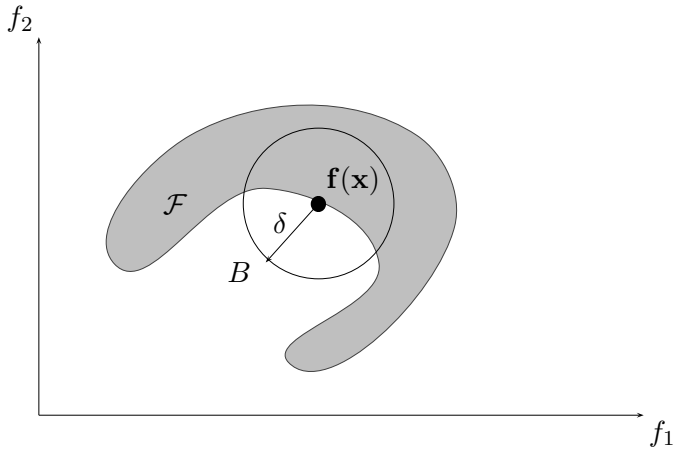


FIG. 1.4 – *Exemple d'optimalité locale.*

Nous avons défini jusqu'ici les notions de *dominance* (au sens de Pareto) et d'optimalité. Il nous reste maintenant à définir la solution d'un problème multiobjectif utilisant ces concepts. Cet ensemble des meilleurs compromis, appelé aussi surface de compromis ou front Pareto, est composé des points qui ne sont dominés par aucun autre. Nous définissons d'abord formellement *l'ensemble des solutions non dominées* :

Définition 11 Ensemble des solutions non dominées : *Soit $\mathcal{F}$ l'image dans l'espace des objectifs de l'ensemble réalisable $\mathcal{X}$. L'ensemble des solutions non dominées de $\mathcal{X}$, est défini par l'ensemble $ND(\mathcal{X})$:*

$$ND(\mathcal{X}) = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{x} \textit{ est non dominé par rapport à } \mathcal{X}\}$$

Nous pouvons définir *le front Pareto* de $\mathcal{F}$ de manière analogue :

Définition 12 Front Pareto : *Soit $\mathcal{F}$ l'image dans l'espace des objectifs de l'ensemble réalisable $\mathcal{X}$. Le front Pareto $ND(\mathcal{F})$ de $\mathcal{F}$ est défini comme suit :*

$$ND(\mathcal{F}) = \{\mathbf{y} \in \mathcal{F} \mid \nexists \mathbf{z} \in \mathcal{F},\ \mathbf{z} < \mathbf{y}\}$$

Le front Pareto est aussi appelé **l'ensemble des solutions efficaces** *ou* **la surface de compromis**

Un exemple de surface de compromis (front Pareto) en dimension 2 est montré à la la figure 1.5. Dans cet exemple, le problème considéré est un problème de minimisation avec deux critères. Deux points particuliers apparaissent clairement : le *point idéal* et le *point Nadir*. Ces deux points sont calculés à partir du front Pareto. Le point idéal (resp. le point Nadir) domine (resp. est dominé par) tous les autres points de la surface de compromis. Bien que ces points ne soient pas forcément compris dans la zone réalisable, ils servent

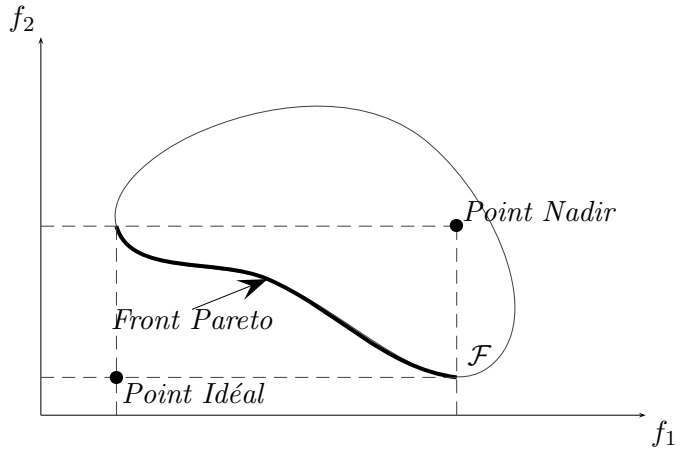


FIG. 1.5 – *Le front Pareto.*

souvent de pôle d'attraction (resp. de répulsion) lors de la résolution du problème. Les coordonnées de ces deux points sont maintenant définies formellement.

Définition 13 Point Idéal : *Les coordonnées du point idéal (pi) correspondent aux meilleurs valeurs de chaque objectif des points du front Pareto. Les coordonnées de ce point correspondent aussi aux valeurs obtenues en optimisant chaque fonction objectif séparément.*

$$pi_i = \min\{\mathbf{y}_i \mid \mathbf{y} \in ND(\mathcal{F})\}$$

Définition 14 Point Nadir : *Les coordonnées du point Nadir (pn) correspondent aux pires valeurs de chaque objectif des points du front Pareto.*

$$pn_i = \max\{\mathbf{y}_i \mid \mathbf{y} \in ND(\mathcal{F})\}$$

La figure 1.5 nous indique aussi que le front Pareto peut avoir des propriétés particulières quant à sa forme. La principale caractéristique utilisée pour comparer les formes de ces courbes est la *convexité*. Nous rappelons donc la définition de la convexité :

Définition 15 Convexité : *Un ensemble A est convexe, si et seulement si l'équivalence suivante est vérifiée :*

$$x \in A \wedge y \in A \Leftrightarrow \mathit{Segment}(x, y) \subset A$$

La convexité est le premier indicateur de la difficulté du problème. En effet, certaines méthodes sont dans l'incapacité de résoudre des problèmes non convexes de manière optimale. Mais il existe d'autres indicateurs tout aussi importants, notamment la continuité, la multimodalité, la nature des variables de décision (entières ou réelles), ...

1.6 Approches de résolution

Un grand nombre d'approches existent pour résoudre les problèmes multiobjectifs. Certaines utilisent des connaissances du problème pour fixer des préférences sur les critères et ainsi contourner l'aspect multicritère du problème. D'autres mettent tous les critères au même niveau d'importance, mais là aussi il existe plusieurs façons de réaliser une telle opération. Plusieurs ouvrages ou articles de synthèse ont été rédigés, des états de l'art plus complets peuvent être consultés notamment dans [Ulungu and Teghem, 1994a; Miettinen, 1999; Ehrgott, 2000; Ehrgott and Gandibleux, 2000; Deb, 2001; Collette and Siarry, 2002].

Parmi toutes ces approches, il faut distinguer deux catégories : les approches *non Pareto* et les approches *Pareto*. Les approches non Pareto ne traitent pas le problème comme un véritable problème multiobjectif. Elles cherchent à ramener le problème initial à un ou plusieurs problèmes mono-objectifs. Par opposition aux méthodes non Pareto, les approches Pareto ne transforment pas les objectifs du problème, ceux-ci sont traités sans aucune distinction pendant la résolution. Nous reviendrons sur ce type d'approches au chapitre 2.

Nous allons dans cette section, décrire les principales approches non Pareto et commenter leurs avantages et leurs inconvénients.

1.6.1 Les méthodes d'agrégation des objectifs

La première approche discutée ici est aussi la plus évidente. Elle consiste à transformer un problème multiobjectif en un problème à un objectif en agrégeant les différents critères sous la forme d'une somme pondérée :

$$\left|\begin{array}{ll} \text{Minimiser} & f_\Sigma = \sum_{i=1}^{m} w_i \cdot f_i(\overrightarrow{x}) \\ \text{tel que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\ \text{avec} & \overrightarrow{x} \in \mathbb{R}^n,\ \overrightarrow{f}(\overrightarrow{x}) \in \mathbb{R}^m,\ \overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^q \end{array}\right.$$

Les w_i, appelés poids, peuvent être normalisés sans perte de généralité : $\sum w_i = 1$. Il est clair que la résolution d'un problème pour un vecteur de poids $\overrightarrow{w}$ fixé ne permet de calculer que quelques solutions Pareto optimales. Pour obtenir un ensemble contenant un grand nombre de solutions Pareto optimales, il faut résoudre plusieurs fois le problème en changeant à chaque fois les valeurs de $\overrightarrow{w}$.

Cette approche a l'avantage évident de pouvoir réutiliser tous les algorithmes classiques dédiés aux problèmes d'optimisation à un seul objectif. C'est souvent la première approche adoptée lorsqu'un chercheur se retrouve devant un nouveau problème multiobjectif.

Cependant cette approche a aussi deux inconvénients importants. Le premier est dû au fait que pour avoir un ensemble de points bien répartis sur le front Pareto, les différents vecteurs $\overrightarrow{w}$ doivent être choisis judicieusement. Il est donc nécessaire d'avoir une bonne connaissance du problème. Le deuxième inconvénient provient du fait que cette méthode ne permet pas, de calculer intégralement la surface de compromis lorsque celle-ci n'est pas convexe. La figure 1.6 illustre ce cas de figure en dimension 2. En effet, pour un vecteur de poids $\overrightarrow{w} = (w_1, w_2)$ fixé, la valeur optimale atteignable pour la fonction objectif créée

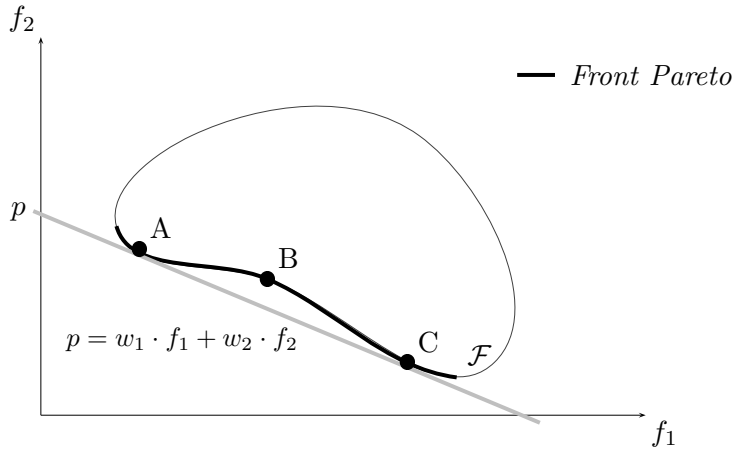


FIG. 1.6 – *Interprétation graphique de l'approche par pondération.*

est p. Les deux points Pareto optimaux trouvés sont A et C. En faisant varier le vecteur $\overrightarrow{w}$, il est possible de trouver d'autres points Pareto optimaux. Seulement, tous ces points se trouveront sur les parties convexes de la surface de compromis. Il n'existe pas de valeur possible pour $\overrightarrow{w}$ permettant de trouver par exemple le point B. En effet, cette approche ne permet pas d'approcher la totalité du front Pareto lorsque celui-ci est non convexe.

1.6.2 L'approche par $\epsilon-$contrainte

Une autre façon de transformer un problème d'optimisation multiobjectif en un problème simple objectif est de convertir $m-1$ des m objectifs du problème en contraintes et d'optimiser séparément l'objectif restant. Le problème peut être reformulé de la manière suivante :

$$
\left|\begin{array}{ll}
\text{Minimiser} & f_i(\overrightarrow{x}) \\
\text{tel que} & f_1(\overrightarrow{x}) \leq \epsilon_1 \\
 & \quad \vdots \\
 & f_{i-1}(\overrightarrow{x}) \leq \epsilon_{i-1} \\
 & f_{i+1}(\overrightarrow{x}) \leq \epsilon_{i+1} \\
 & \quad \vdots \\
 & f_m(\overrightarrow{x}) \leq \epsilon_m \\
\text{et que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\
\text{avec} & \overrightarrow{x} \in \mathbb{R}^n,\ \overrightarrow{f}(\overrightarrow{x}) \in \mathbb{R}^m,\ \overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^q
\end{array}\right.
$$

L'approche par $\epsilon-$contrainte doit aussi être appliquée plusieurs fois en faisant varier le vecteur $\overrightarrow{\epsilon}$ pour trouver un ensemble de points Pareto optimaux.

Cette approche a l'avantage par rapport à la précédente de ne pas être trompée par les problèmes non convexes. Ainsi la figure 1.7 illustre, en dimension 2, le cas où un point $(\epsilon, f_{1_{min}})$, de la partie non convexe, est trouvé. La figure 1.7 montre aussi comment cette

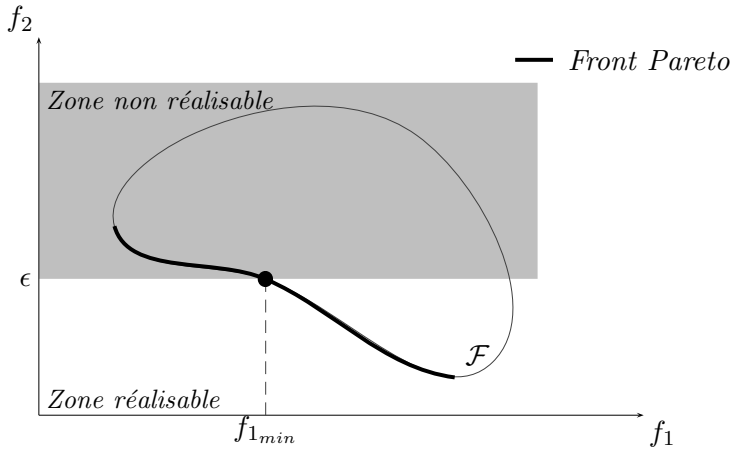


FIG. 1.7 – *Interprétation graphique de l'approche par $\epsilon-$contrainte.*

approche procède. En transformant des fonctions objectifs en contraintes, elle diminue la zone réalisable par paliers. Ensuite, le processus d'optimisation trouve le point optimal sur l'objectif restant.

L'inconvénient de cette approche réside dans le fait qu'il faille lancer un grand nombre de fois le processus de résolution. De plus, pour obtenir des points intéressants et bien répartis sur la surface de compromis, le vecteur $\overrightarrow{\epsilon}$ doit être choisi judicieusement. Il est clair qu'une bonne connaissance du problème *a priori* est requise.

1.6.3 L'approche Min-Max

Cette approche consiste à transformer le problème multiobjectif en un problème à un seul objectif où l'on cherche à minimiser l'écart relatif par rapport à un point de référence appelé *but*, fixé par la méthode ou le décideur. Il existe plusieurs manières de caractériser la distance entre un point de référence (le but) et un autre, notamment à l'aide de normes. Une norme est définie de la manière suivante :

$$L_r(\overrightarrow{f}(\overrightarrow{x})) = \left[\sum_{i=1}^{m} |B_i - f_i(\overrightarrow{x})|^r\right]^{\frac{1}{r}}$$

Les principales normes utilisées sont : $L_1 = \sum_{i=1}^{m} |B_i - f_i(\overrightarrow{x})|$, la distance classique, et la norme $L_\infty = \max_{i\in\{1,\cdots,m\}}(B_i - f_i(\overrightarrow{x}))$. C'est cette dernière qui est utilisée dans l'approche min-max appelée aussi approche de *Tchebychev* [Miettinen, 1999] :

Minimiser	$\max_{i\in\{1,\cdots,m\}}(B_i - f_i(\overrightarrow{x}))$
tel que	$\overrightarrow{g}(\overrightarrow{x}) \leq 0$
avec	$\overrightarrow{x} \in \mathbb{R}^n,\ \overrightarrow{f}(\overrightarrow{x}) \in \mathbb{R}^m,\ \overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^q$

Dans cette approche, le point de référence joue un rôle fondamental. S'il est mal choisi, la recherche peut s'avérer être très laborieuse. La figure 1.8 illustre, en dimension 2, le

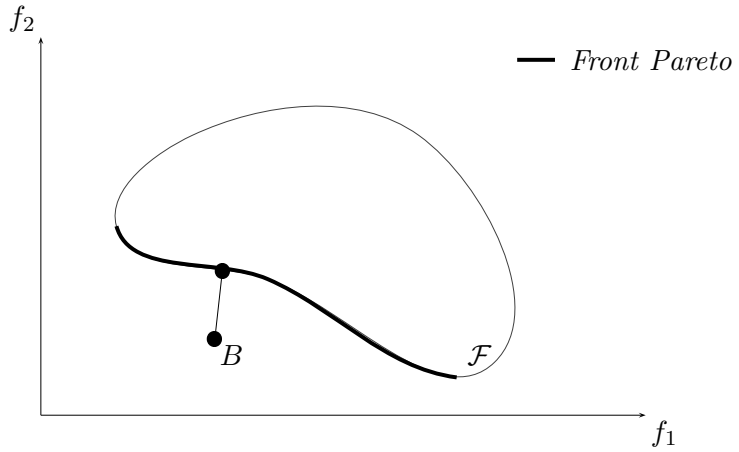


FIG. 1.8 – *Interprétation graphique de l'approche Min-Max.*

cas d'une recherche avec un but B fixé. Il est clair que l'approche permet de traiter les problèmes non convexes à condition que le point de référence soit choisi judicieusement. Les méthodes de résolution implémentant cette approche utilisent souvent le *point idéal* comme point de référence. Ce point idéal évolue donc en fonction de la recherche. En effet, plus la surface de compromis courante trouvée par la méthode se rapproche du front Pareto optimal, plus le point idéal se rapprochera du point idéal du problème.

1.6.4 Le but à atteindre

Cette approche, comme celle de min-max, utilise un point de référence pour guider la recherche. Mais elle introduit aussi une direction de recherche, si bien que le processus de résolution devra suivre cette direction. À la différence de l'approche min-max, qui utilise des normes pour formaliser la distance au point de référence, l'approche du *but à atteindre* utilise des contraintes, à l'instar de l'approche $\epsilon-$contrainte, pour déterminer la position du point de référence (aussi appelé le but). L'écart par rapport à ce but est contrôlé grâce à la variable λ introduite à cet effet :

$$
\left|\begin{array}{ll}
\text{Minimiser} & \lambda \\
\text{tel que} & f_1(\overrightarrow{x}) - w_1 \cdot \lambda \leq B_1 \\
 & \quad \vdots \\
 & f_m(\overrightarrow{x}) - w_m \cdot \lambda \leq B_m \\
\text{et que} & \overrightarrow{g}(\overrightarrow{x}) \leq 0 \\
\text{avec} & \overrightarrow{x} \in \mathbb{R}^n,\ \overrightarrow{f}(\overrightarrow{x}) \in \mathbb{R}^m,\ \overrightarrow{g}(\overrightarrow{x}) \in \mathbb{R}^q
\end{array}\right.
$$

Ainsi en minimisant λ et en vérifiant toutes les contraintes, la recherche va s'orienter vers le but B et s'arrêter sur le point A faisant partie de la surface de compromis (voir l'illustration en 2 dimensions à la figure 1.9). Cette approche permet, comme l'approche par ϵ-contrainte et l'approche min-max, de trouver les parties non convexes des fronts Pareto.

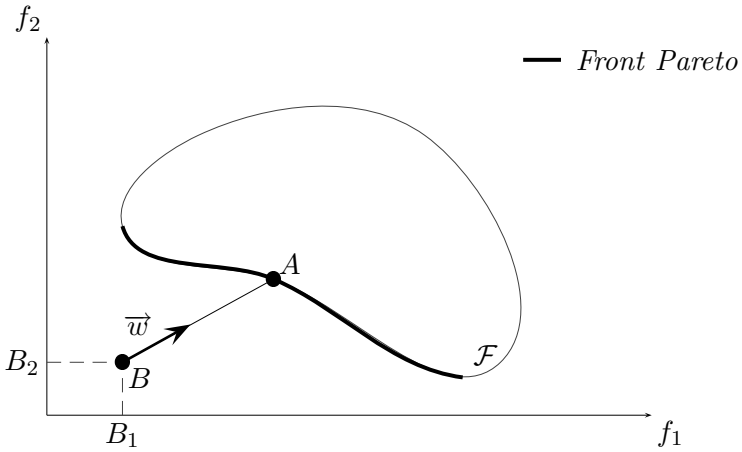


FIG. 1.9 – *Interprétation graphique de l'approche par "but à atteindre".*

Cependant, cette approche, comme les précédentes, doit être itérée plusieurs fois dans le but d'obtenir un ensemble de points Pareto optimaux. Les paramètres $\overrightarrow{w}$ et B doivent être bien choisis par l'utilisateur. Bien que ces paramètres permettent une grande flexibilité de la recherche (orientation et but), s'ils sont mal choisis, ils peuvent, dans certains cas extrêmes, donner des résultats non cohérents.

Il est à noter que cette approche est très similaire à celle de la "goal programming" [Charnes and Cooper, 1961; Romero, 1991; Coello, 1998], où les contraintes deviennent des égalités. Des variables d'écart sont alors introduites.

1.6.5 Discussion

Nous avons présenté dans ce chapitre les principaux concepts de l'optimisation multiobjectif. Pour ce chapitre et pour le reste de cette thèse, nous nous plaçons dans le cadre de problèmes d'optimisation où il n'existe pas de modèle de préférences sur les critères (tous les critères sont de même importance). Toutes les problématiques liées à la modélisation des préférences, au choix de solutions au sein d'un ensemble de compromis, ou à la résolution interactive, sont traitées par *la communauté d'aide à la décision dans un contexte multicritère*. Le lecteur intéressé pourra consulter [Vincke, 1992; Bouyssou *et al.*, 2001] pour une présentation plus détaillée de ces problématiques.

Les différentes approches de résolution présentées dans ce chapitre sont des approches non Pareto. Elles transforment un problème d'optimisation multiobjectif en un ou plusieurs problèmes à un seul objectif. Que ce soit sous la forme d'une somme pondérée, ou sous la forme d'une distance à un but, cette transformation permet d'utiliser facilement les méthodes d'optimisation issues de l'optimisation à un objectif.

Mais ces méthodes ont aussi des inconvénients. Certaines ne peuvent traiter complètement des problèmes non convexes et sont donc très sensibles à la forme du front Pareto (*somme pondérée, ...*). Les autres, bien que pouvant traiter les problèmes non convexes, restent quand même sensibles à la forme du front Pareto (*min-max, but à atteindre, ...*).

Un autre inconvénient important est qu'il faille relancer plusieurs fois les algorithmes de résolution avec des valeurs différentes pour certains paramètres (vecteur de poids par exemple) pour obtenir plusieurs points distincts de la surface de compromis. Ces méthodes nécessitent aussi souvent une bonne connaissance du problème a priori, notamment pour fixer les vecteurs de poids ou les points de référence.

Récemment les métaheuristiques, notamment les algorithmes évolutionnaires, ont permis la réalisation de méthodes de résolution dites Pareto très performantes. Elles permettent de déterminer en une seule exécution une approximation de l'intégralité du front Pareto, et ceci même si les problèmes sont non convexes. Le chapitre suivant est dédié à cette classe de méthodes relativement récentes.

Chapitre 2

Les métaheuristiques pour l'optimisation multiobjectif

Dans ce chapitre, nous présentons quelques méthodes de résolution Pareto fondées sur des métaheuristiques. Ainsi, nous introduisons les méthodes de recherche locale et les approches évolutionnaires. Nous décrivons aussi deux algorithmes évolutionnaires représentatifs adoptant une approche Pareto pour résoudre les problèmes d'optimisation multiobjectifs.

Sommaire

2.1 Introduction

Tout comme pour l'optimisation mono-objectif, on peut distinguer deux grandes familles de méthodes de résolution pour traiter un problème multiobjectif : les méthodes exactes et les méthodes approchées.

On remarque tout abord qu'il y a très peu de travaux sur les méthodes exactes dans le contexte de la résolution des problèmes d'optimisation multiobjectifs NP-difficiles, sans doute, à cause de la grande difficulté de ce type de problème. Les références [Ulungu, 1993; Ulungu and Teghem, 1994b; Visée *et al.*, 1998; Ehrgott and Gandibleux, 2000] présentent la plupart des méthodes exactes existantes. Dans cette thèse, nous nous intéressons à la résolution approchée de problèmes d'optimisation multiobjectifs NP-difficiles, notamment par des métaheuristiques. Ainsi, nous présentons brièvement dans ce chapitre deux types de métaheuristiques les plus connues : la recherche locale [1] et les algorithmes évolutionnaires.

Pour une présentation plus élaborée des métaheuristiques, le lecteur est invité à consulter les références suivantes [Reeves, 1995; Hao *et al.*, 1999; Corne *et al.*, 1999; Pirlot and Teghem, 2002; Dréo *et al.*, 2003]. Les métaheuristiques ont été appliquées avec succès sur un grand nombre de problèmes académiques et réels : problème d'affectation quadratique, coloriage de graphe, voyageur de commerce, ... Le lecteur intéressé peut consulter, par exemple, [Pirlot and Teghem, 2003] pour une présentation de quelques applications importantes.

2.2 Les méthodes de recherche locale

La première classe de métaheuristiques présentées regroupe les méthodes utilisant les principes de la recherche locale. Ces méthodes résolvent le problème d'optimisation de manière itérative. Elles font évoluer la configuration courante en la remplaçant par une autre issue de son voisinage, ce changement de configuration est couramment appelé un mouvement.

2.2.1 Déterminer le voisinage

Toutes les approches de recherche locale utilisent la notion de voisinage. Un aspect fondamental de ces approches est donc la détermination de ce voisinage.

Déterminer le voisinage consiste à caractériser tous ses éléments. Le voisinage est souvent représenté par une fonction $\mathcal{N}$ qui, à un point $\mathbf{x}$, associe un ensemble de points $\mathcal{N}(\mathbf{x})$. Il existe une infinité de manières de choisir $\mathcal{N}$, il faut adapter ce choix au problème, c'est-à-dire choisir la meilleure fonction $\mathcal{N}$ selon le problème considéré.

Définition 16 Fonction de voisinage : *Soit $\mathcal{X}$ l'espace de recherche d'un problème. Une fonction de voisinage $\mathcal{N}$ est une association $\mathcal{N} : \mathcal{X} \longrightarrow 2^{\mathcal{X}}$, définissant, pour chaque point $\mathbf{x} \in \mathcal{X}$, un ensemble $\mathcal{N}(\mathbf{x}) \subseteq \mathcal{X}$ de points "proches" de $\mathbf{x}$.*

1. Notons que dans la littérature, le terme "recherche locale" est de plus en plus employé pour désigner non seulement la méthode de *descente*, mais plus généralement les méthodes à voisinage. Nous utilisons ce terme dans ce sens.

Un autre aspect important est la taille du voisinage. En effet, certaines fonctions $\mathcal{N}$ peuvent calculer un ensemble si grand qu'il est impossible de le traiter efficacement avec un ordinateur (dépassement de la taille mémoire par exemple). Il convient alors soit de changer la fonction $\mathcal{N}$, soit, pour un voisinage complètement différent, soit, et c'est souvent la solution retenue, pour restreindre le voisinage à un ensemble contenant moins d'éléments.

Il est clair que plusieurs points $\mathbf{x}$ seront visités pendant la recherche. Certaines méthodes comme le *recuit simulé* ne calculent pas tout le voisinage mais uniquement certains points. D'autres, comme la *recherche Tabou*, préconisent l'examen de tous les voisins pour réaliser un mouvement. Dans ce dernier cas de figure, il est important que le voisinage $\mathcal{N}(\mathbf{x})$ soit calculé rapidement. L'évaluation du ou des voisins est un autre aspect important de ces méthodes. En effet, le choix de la prochaine configuration dépend en grande partie de l'évaluation des voisins. Plus le voisinage est important et plus les phases d'évaluations sont longues. Il est donc impératif que l'évaluation d'un candidat soit très rapide. Le plus souvent les méthodes de recherche locale utilisent des fonctions d'évaluation dites incrémentales. Ces fonctions se basent sur le changement local de $\mathbf{x}$ pour calculer de manière incrémentale la nouvelle évaluation. En effet, elles permettent, si un ordre de parcours du voisinage est respecté, d'évaluer chaque voisin très rapidement.

2.2.2 La descente

La descente est une méthode d'amélioration itérative simple permettant d'atteindre le premier optimum local. La descente pour un problème de minimisation peut être définie très simplement :

Algorithme 1 Pseudo-code de la méthode de descente.

```
Paramètres d'entrée: 𝒩, f, x_0
x_suiv ← x_0
Répeter
    x ← x_suiv
    x_suiv ∈ {x' | x' ∈ 𝒩(x) ∧ f(x') = min({f(y) | y ∈ 𝒩(x)})}
Jusqu'à  f(x_suiv) > f(x)
Renvoyer x
```

où $\mathcal{N}$ est la fonction de voisinage, f la fonction d'évaluation, et $\mathbf{x_0}$ la configuration initiale servant de point de départ à l'algorithme.

Ainsi le plus proche optimum local de $\mathbf{x_0}$ est trouvé. Mais celui-ci peut être loin de l'optimum global, et être une mauvaise approximation de cet optimum. Pour essayer de se rapprocher de l'optimum global, plusieurs techniques sont envisageables : la première technique couramment utilisée est celle de la relance. Elle consiste à recommencer une nouvelle recherche à partir d'un autre point initial, si possible loin du précédent. La deuxième technique est celle du *chemin aléatoire*. Elle consiste à effectuer, de temps en temps, un mouvement aléatoire pour diversifier la recherche et ainsi espérer se rapprocher de l'optimum global.

La descente est largement utilisée, et est souvent la première méthode expérimentée sur un nouveau problème. Elle permet, dans un temps de développement assez court, de bien appréhender le problème et de calculer rapidement des premières approximations de l'optimum global.

2.2.3 Le recuit simulé

Le *recuit simulé* [Kirkpatrick *et al.*, 1983] s'inspire du processus de recuit physique. Le processus du recuit simulé répète une procédure itérative qui cherche des configurations de coût plus faible tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût.

Nous donnons maintenant le pseudo-code d'un algorithme type du recuit simulé pour un problème de minimisation :

Algorithme 2 Pseudo-code du recuit simulé.

```
Paramètres d'entrée: T_0, Seuil, α, it_palier, N, f, x_0
x ← x_0
T ← T_0
Tant que T > Seuil faire
    nombre_itérations ← 0
    Tant que nombre_itérations < it_palier faire
        nombre_itérations ← nombre_itérations + 1
        Choisir x' ∈ N(x)
        Δf ← f(x') − f(x)
        Si Δf < 0 Alors
            x ← x'
        Sinon
            Tirer r de manière aléatoire dans l'intervalle [0, 1]
            Si r < e^(−Δf/T) Alors
                x ← x'
            FinSi
        FinSi
    FinTantQue
    T ← α(T)
FinTantQue
Renvoyer x
```

où $\mathcal{T}_0$ est la température initiale, *Seuil* le seuil minimal que la température peut atteindre, α la fonction diminuant la température à certains paliers, it_{palier} le nombre d'itérations à effectuer dans un palier, $\mathcal{N}$ la fonction de voisinage, f la fonction d'évaluation, et $\mathbf{x_0}$ la configuration initiale servant de point de départ à l'algorithme. Dans l'algorithme que nous présentons, le nombre d'itérations (it_{palier}) devant être atteint pour effectuer un changement de palier est fixe. Dans la pratique, les algorithmes de recuit simulé font

varier ce paramètre en fonction de la température actuelle. Il est possible, par exemple, d'augmenter cette valeur lorsque la température diminue. De même, la fonction α diminuant la température pendant la recherche peut tenir compte d'autres paramètres non indiqués ici. Par exemple, beaucoup d'algorithmes de recuit simulé font intervenir le nombre d'itérations ou le taux d'amélioration de la solution courante pour calculer la diminution de la température.

Un présentation détaillée de cette méthode est donnée dans [Aarts and Korst, 1989]. Le recuit simulé a acquis son succès notamment grâce à des résultats pratiques obtenus sur de nombreux problèmes dans des domaines variés. Des exemples de ces applications sont présentés dans [Aarts and Korst, 1989; Vidal, 1993; Koulamas *et al.*, 1994].

2.2.4 La recherche Tabou

La recherche Tabou est introduite par Glover [Glover, 1986]. Cette méthode est amplement présentée dans [Glover and Laguna, 1997].

Son fonctionnement est plus proche de la méthode de descente (Section 2.2.2) que du recuit simulé (Section 2.2.3). La recherche Tabou examine un échantillonnage de configurations de $\mathcal{N}(\mathbf{x})$ et retient la meilleure $\mathbf{x}'$ même si $\mathbf{x}'$ est plus mauvaise que $\mathbf{x}$. Cependant, cette stratégie peut entraîner des cycles, par exemple le cycle de longueur 2 : $\mathbf{x} \rightarrow \mathbf{x}' \rightarrow \mathbf{x} \rightarrow \mathbf{x}' \rightarrow \cdots$. Pour empêcher ce type de cycle d'apparaître, on mémorise les k dernières configurations visitées dans une mémoire à court terme et on interdit de retenir toute configuration qui en fait déjà partie. Cette mémoire, appelée la *liste Tabou*, est une des composantes essentielles de la méthode. En effet, elle permet d'éviter tous les cycles de longueur inférieure à k, k étant déterminée en fonction du problème. Plusieurs algorithmes de recherche Tabou sont présentés à la Section 4.2.

La mémorisation de configurations entières serait trop coûteuse en temps de calcul et en place mémoire. Il est préférable de mémoriser des *caractéristiques* des configurations au lieu de configurations entières. Plus précisément, il est possible de mémoriser uniquement un attribut de la configuration. Il en résulte que toutes les configurations possédant cet attribut, y compris celles qui n'ont pas encore été rencontrées, deviennent elles aussi Tabou. Pour palier à ce problème, un mécanisme particulier, appelé *l'aspiration*, est mis en place. Ce mécanisme consiste à révoquer le statut Tabou d'une configuration à certains moments de la recherche. La fonction d'aspiration la plus simple consiste à enlever le statut Tabou d'une configuration si celle-ci est meilleure que la meilleure configuration trouvée jusqu'alors.

Il existe aussi d'autres techniques permettant d'améliorer les performances de la méthode Tabou, en particulier, l'*intensification* et la *diversification*. L'intensification permet de se focaliser sur certaines zones de l'espace de recherche en apprenant des propriétés favorables, par exemple les propriétés communes souvent rencontrées dans les meilleurs configurations visitées. La diversification a un objectif inverse de l'intensification. En effet, elle cherche à diriger la recherche vers des zones inexplorées, en modifiant par exemple la fonction d'évaluation. L'intensification et la diversification jouent donc des rôles complémentaires.

La recherche Tabou a fait l'objet de bien des développements ces dernières années. Elle

est utilisée pour traiter un grand nombre de problèmes d'optimisation : coloriage de graphe, sac à dos, sac à dos multidimensionnel .... La majorité des développements apportés ainsi qu'un large éventail d'applications de la recherche Tabou, se retrouvent dans [Glover and Laguna, 1997].

2.3 Les algorithmes évolutionnaires

La deuxième classe de métaheuristiques présentée dans cette thèse est celle des *algorithmes évolutionnaires* [Back *et al.*, 1997]. On peut distinguer trois grandes classes d'algorithmes évolutionnaires : les algorithmes génétiques [Holland, 1975; Goldberg, 1989], les stratégies d'évolution [Schwefel, 1981] et la programmation évolutive [Fogel, 2000]. Ces méthodes se différencient par leur manière de représenter l'information et par leur façon de faire évoluer la population d'une génération à l'autre. Un algorithme évolutionnaire est typiquement composé de trois éléments fondamentaux :

- une *population* constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné,
- un *mécanisme d'évaluation* des individus permettant de mesurer l'adaptation de l'individu à son environnement,
- un *mécanisme d'évolution* de la population permettant, grâce à des opérateurs prédéfinis, d'éliminer certains individus et d'en créer de nouveaux.

Parmi les composants d'un algorithme évolutionnaire, l'individu et la fonction d'évaluation correspondent respectivement à la notion de configuration et à la fonction d'évaluation dans les méthodes de voisinage. Le mécanisme d'évolution est composé de plusieurs opérateurs tels que la sélection, la mutation et le croisement.

La *sélection* a pour objectif de sélectionner des individus qui vont pouvoir se reproduire pour transmettre leurs caractéristiques à la génération suivante. Le *croisement* ou recombinaison est un opérateur permettant de construire des nouveaux individus enfants à partir des caractéristiques d'individus parents sélectionnés. La *mutation* effectue des légères modifications de certains individus.

Comme exemple des algorithmes évolutionnaires, nous présentons maintenant les algorithmes génétiques.

2.3.1 Les algorithmes génétiques

Les algorithmes génétiques (AGs) ont été introduits par Holland [Holland, 1975] comme un modèle de méthode adaptative. Ils s'appuient sur un codage de l'information sous forme de chaînes binaires de longueur fixe et d'un ensemble d'opérateurs génétiques : la sélection, la mutation, le croisement, ... Un individu sous ce codage, appelé un chromosome, représente une configuration du problème.

La figure 2.1 donne un exemple de représentation de l'information stocké dans un chromosome codé sur 8 bits. L'évaluation de cet individu consiste à transformer la chaîne 0/1

du chromosome en une valeur réelle, appelée valeur d'adaptation. La fonction d'évaluation dépend principalement de l'objectif du problème. Si cette fonction est uniquement basée sur la fonction objectif du problème, on utilisera le terme de *fonction de coût* pour la désigner.

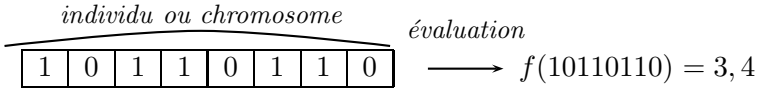


FIG. 2.1 – *Codage de l'information.*

La valeur d'adaptation obtenue pour chaque chromosome lors de l'évaluation est utilisée, notamment lors des opérations de *sélection*, pour choisir les individus amener à se reproduire par des croisements ou à être utilisés par d'autres opérateurs génétiques.

Le *croisement* permet de produire deux nouveaux individus, appelés les enfants, à partir de deux individus, appelés les parents. La figure 2.2 montre un exemple de croisement monopoint, qui consiste à échanger les segments des deux parents déterminés par le point de croisement.

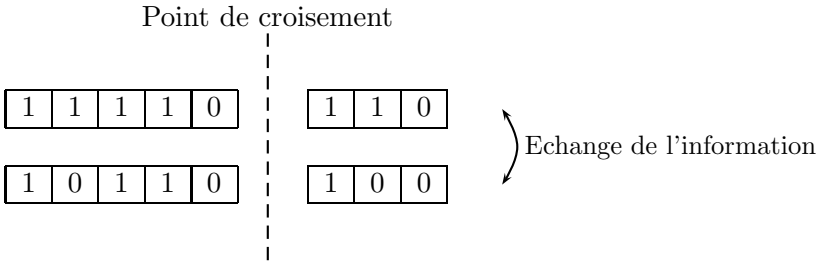


FIG. 2.2 – *Opérateur de croisement monopoint appliqué à deux chromosomes codés sur* 8 *bits.*

La *mutation* consiste à changer la valeur de certaines variables du chromosome. Les variables à modifier sont le plus souvent choisies aléatoirement. La figure 2.3 montre un exemple de mutation sur un chromosome codé sur 8 bits.

Ainsi, la nouvelle population construite à partir des opérateurs génétiques présentés devient la population de référence de la prochaine génération. Ce cycle d'opérations continue tant que la méthode n'a pas rencontré une condition d'arrêt définie préalablement, un nombre maximal de générations par exemple.

Les algorithmes génétiques, et plus généralement les algorithmes évolutionnaires ont permis de résoudre un grand nombre de problèmes, notamment en optimisation. Dans le contexte multiobjectif, l'approche évolutionnaire offre, par le biais de la notion de population, des mécanismes pertinants pour approcher la solution optimale (front Pareto).

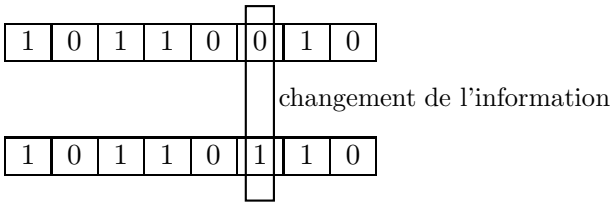


FIG. 2.3 – *Opérateur de mutation appliqué à deux chromosomes codés sur* 8 *bits.*

Nous présentons maintenant trois de ces mécanismes : une sélection Pareto, l'élitisme et la diversification.

2.3.2 Un mécanisme de sélection Pareto

Un des principaux avantages des algorithmes évolutionnaires pour l'optimisation multiobjectif est qu'ils permettent non seulement la mise en oeuvre d'approches non Pareto (agrégation des objectifs, ...), mais aussi l'implémentation d'approches Pareto. En effet, certains mécanismes de sélection implémentent la relation de dominance (voir Section 1.4) réalisant ainsi une sélection Pareto.

Une sélection Pareto utilise la relation de dominance pour affecter des rangs aux individus de la population, faisant apparaître la notion de front. Citons par exemple la technique de *ranking*, dont l'idée, suggérée dans [Goldberg, 1989], fut reprise et implémentée dans l'algorithme *NSGA* [Srinivas and Deb, 1994]. Cet algorithme, toujours très populaire dans la communauté **EMOO** [2], sert encore de base pour le développement de nouveaux algorithmes.

Goldberg présenta dans ses travaux une procédure itérative de seulement 10 lignes pour calculer ce rang : initialement, tous les individus non dominés de la population reçoivent le rang 1 et sont retirés temporairement de la population. Puis, les nouveaux individus non dominés reçoivent le rang 2 avant d'être à leur tour retirés. Le processus s'itère tant qu'il reste des individus dans la population. La valeur d'adaptation de chaque individu correspond à son rang dans la population. Ainsi, l'évaluation d'un individu ne dépend pas uniquement de lui même, mais aussi de la population (cf. Figure 2.4).

Ce principe a l'avantage de ne pas hiérarchiser les objectifs entre eux. Mais l'augmentation de la taille de l'espace de recherche (surtout créée par l'augmentation du nombre de variables) peut influencer la performance de cette sélection. En effet, comment sélectionner certains individus si tous ont le même rang ! Ce cas peut arriver d'autant plus que le nombre de variables est grand, et que le problème est multimodal.

2. EMOO : Evolutionary Multi Objective Optimization

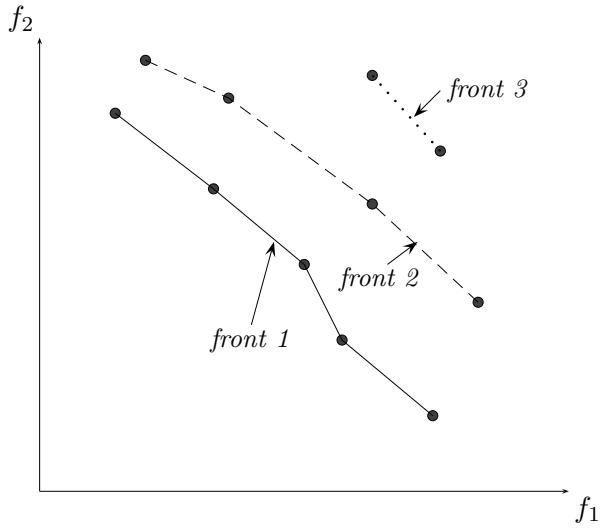


FIG. 2.4 – *Le "ranking".*

2.3.3 L'élitisme

L'élitisme permet de conserver les meilleurs individus dans les générations futures. Une des premières implémentations de ce mécanisme dans un algorithme génétique est présentée dans [De Jong, 1975].

L'élitisme est introduit pour conserver les bonnes solutions lors du passage de la génération courante à la prochaine génération. Conserver ces solutions pour les générations futures permet d'améliorer les performances des algorithmes sur certains problèmes. En effet, dans ses expérimentations, De Jong a observé qu'un algorithme génétique élitiste améliorait significativement les résultats sur des fonctions unimodales. D'un autre côté, l'élitisme peut causer une convergence prématurée sur des fonctions multimodales.

Réaliser un algorithme élitiste dans le cadre des problèmes multiobjectifs est plus difficile que pour les problèmes à un objectif. En effet, la meilleure solution n'est plus un unique individu, mais tout un ensemble dont la taille peut aller jusqu'à dépasser la taille maximale de la population. Deux adaptations du mécanisme élitiste sont considérées : la première approche regroupe les algorithmes, fondés sur les travaux de De Jong, qui conservent pour les générations futures les k meilleurs individus [Tamaki *et al.*, 1994; Deb *et al.*, 2000]. Mais comment avec cette approche sélectionner k individus, si l'ensemble des points non dominés actuel comporte plus de k solutions ? Il y a un risque de perdre une partie du front Pareto optimal, et le concept de l'élitisme n'est plus complètement présent. Les approches récentes [Ishibuchi and Murata, 1996; Parks and Miller, 1998; Zitzler and Thiele, 1999] tendent à utiliser une population externe d'individus dans laquelle est stocké le meilleur ensemble des points non dominés découverts jusqu'ici. Cet ensemble est mis à jour continuellement pendant la recherche, et les individus stockés continus à pouvoir être choisis par l'opérateur de sélection. Ils peuvent ainsi se reproduire et transmettre leurs caractéristiques aux générations suivantes.

Actuellement, les algorithmes élitistes obtiennent de meilleurs résultats sur un grand nombre de problèmes multiobjectifs [Zitzler and Thiele, 1999; Deb *et al.*, 2001].

2.3.4 Maintenir la diversité !

La diversité est une notion déjà importante lors de l'optimisation de problèmes à un objectif, elle devient prépondérante lorsque l'on traite de problèmes multiobjectifs. Maintenir un certain degré de diversité dans la population d'un algorithme évolutionnaire consiste à éviter que la population ne converge prématurément vers une petite zone de l'espace de recherche ou de l'espace des objectifs. En effet, si il n'existe pas de mécanisme de contrôle de la diversité, les opérations de sélection vont privilégier *trop vite* certains individus meilleurs à cette étape de la recherche. Cette convergence prématurée a comme effet de limiter la recherche à un sous-ensemble plus restreint de l'espace de recherche, qui peut ne contenir aucune solution optimale. Dans le cas de problèmes multiobjectifs, converger prématurément rend impossible la découverte de l'intégralité du front Pareto. En effet, les individus de la population se focaliseront sur une partie du front Pareto, et ne se répartiront pas sur la totalité de ce front Pareto.

Pour remédier à ce problème, de nombreuses techniques ont été développées. Celles-ci influent sur la *pression de sélection*, afin de privilégier certains individus, permettant d'obtenir une population plus diversifiée. Cependant, ces techniques ajoutent un coût calculatoire non négligeable pour l'algorithme, elles doivent donc être choisies avec soin. Nous allons maintenant présenter les mécanismes de diversification les plus couramment intégrés aux algorithmes évolutionnaires.

Le "sharing"

Le *sharing* consiste à modifier la valeur de coût d'un individu (calculée uniquement à partir de la fonction objectif du problème). C'est cette nouvelle valeur qui sera utilisée comme valeur d'adaptation par l'opérateur de sélection. Cette technique, introduite dans [Goldberg and Richardson, 1987], est largement utilisée aujourd'hui.

Pour éviter qu'un trop grand nombre d'individus ne se concentrent autour d'un même point, il faut pénaliser la valeur d'adaptation en fonction du nombre d'individus au voisinage du regroupement : plus les individus sont regroupés, plus leur valeur d'adaptation est faible, et des individus proches les uns des autres doivent partager leur valeur d'adaptation. Dans la pratique, on estime ce taux de concentration en ouvrant un *domaine* autour d'un individu, puis on calcule les distances entre les individus contenus dans ce domaine.

Pour déterminer les bornes du domaine ouvert autour de l'individu choisi, on définit une distance maximale, appelée σ_{share}, au delà de laquelle les individus ne seront plus considérés comme faisant parti du domaine ouvert. La distance séparant deux individus i et j est calculée grâce à la fonction $d(i,j)$. La valeur d'adaptation $F(i)$ d'un individu $i \in P(population)$ est égale à son coût $F'(i)$ divisé par sa *valeur de niche* :

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

où la fonction Sh est définie comme suit :

$$Sh(d(i,j)) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{share}}\right)^2 & \text{si } d(i,j) < \sigma_{share} \\ 0 & \text{sinon} \end{cases}$$

La fonction $d(i, j)$ de calcule de distance peut être définie dans l'espace de recherche, par exemple à l'aide d'une distance de Hamming, ou dans l'espace objectif. Ce choix dépend souvent du problème, car le maintient de la diversité dans l'espace objectif, bien qu'il soit souvent plus simple à réaliser, n'assure pas forcément le maintient de la diversité dans l'espace de recherche.

La réinitialisation

La réinitialisation est une technique largement utilisée par toutes les métaheuristiques, les algorithmes évolutionnaires n'échappant pas à la règle. Lors d'approches par population, elle consiste à réinitialiser un certain nombre d'individus de la population, par exemple de manière aléatoire. En introduisant de manière régulière certains individus générés aléatoirement, de nouvelles zones de l'espace de recherche, peut-être inexplorées jusqu'ici, peuvent être découvertes.

Le "crowding"

L'approche par "crowding" consiste à déterminer un représentant par niche découverte. À la différence du "sharing", où tous les individus sont susceptibles d'être sélectionnés et de participer aux phases de croisement, mutation et sélection, avec le "crowding", seuls les représentants participeront aux différentes étapes de l'algorithme. Le "crowding" fut introduit par De Jong [De Jong, 1975] et fut adapté notamment au travers des travaux de Blickle [Blickle, 1996].

2.3.5 Les méthodes hybrides

Une autre façon d'améliorer les performances d'un algorithme ou de combler certaines de ses lacunes consiste à le combiner avec une autre méthode [Talbi, 2000]. Ce principe général, appelé hybridation, peut s'appliquer pour un grand nombre de méthodes. Les algorithmes évolutionnaires ne font pas exception à la règle, et une multitude d'algorithmes hybrides ont fait leur apparition ces dernières années. Un cas particulier de l'hybridation entre deux méthodes consiste à combiner un algorithme génétique et une méthode de recherche locale.

Dans une telle hybridation, on substitue souvent la mutation par une méthode de recherche locale. En effet, les méthodes de recherche locale remplacent la configuration courante par une autre voisine. Il y a donc peu de modifications séparant les deux configurations. L'opérateur de mutation des algorithmes évolutionnaires effectue lui aussi des modifications légères sur la configuration sélectionnée. De manière intuitive, un grand nombre de chercheurs ont donc substitué une méthode de recherche locale à l'opérateur de mutation, pour créer une méthode hybride. Nous pouvons ainsi citer pour le cas des problèmes multiobjectifs : la méthode *MOTS* [Hansen, 1997; Gandibleux *et al.*, 1997] combinant une population et une recherche Tabou, la méthode *PSA* [Czyzak and Jaszkiewicz, 1998] combinant un algorithme génétique et le recuit simulé, la méthode *M-PAES* [Corne and Knowles, 2000] intégrant un schéma généralisant l'implémentation d'un grand nombre d'algorithmes hybrides pour l'optimisation des problèmes multiobjectifs.

2.4 Quelques algorithmes évolutionnaires performants

Récemment, beaucoup de recherches ont été menées sur l'application des algorithmes évolutionnaires aux problèmes d'optimisation multiobjectifs. Celles-ci ont permis de mettre en avant l'intérêt d'utiliser des méthodes d'optimisation basées sur le concept de population [Goldberg, 1989; Fonseca and Fleming, 1993; Srinivas and Deb, 1994; Hansen, 1997; Gandibleux *et al.*, 1997; Zitzler and Thiele, 1999; Corne and Knowles, 2000; Coello *et al.*, 2002]. Nous présentons maintenant deux algorithmes évolutionnaires représentatifs, résolvant des problèmes d'optimisation multiobjectifs. Ces deux méthodes ont donné lieu à des travaux récents et nombreux [Zitzler and Thiele, 1999; Zitzler *et al.*, 2000; Deb and Goel, 2001], illustrant leurs originalités et leurs bonnes performances sur de nombreuses instances de problèmes.

2.4.1 Le "Non Sorting Dominated Genetic Algorithm II" (*NSGA-II*)

Srinivas et Deb en 1994 [Srinivas and Deb, 1994] ont réalisé une implémentation quasi directe du schéma de sélection introduit par Goldberg [Goldberg, 1989], nommé le *Non Sorting Genetic Algorithm* (*NSGA*). Les différents fronts de compromis de la population sont recueillis un par un, et la valeur d'adaptation est calculée sur chaque front de manière indépendante, préservant ainsi la diversité. Une nouvelle version élitiste de cet algorithme, nommée *Non Sorting Genetic Algorithm II* (*NSGA-II*), a été présentée dans [Deb *et al.*, 2000]. *NSGA-II* intègre un opérateur de sélection, basé sur un calcul de la distance de "crowding", très différent de celui de *NSGA*. Comparativement à *NSGA*, *NSGA-II* obtient de meilleurs résultats sur toutes les instances présentées dans les travaux de K. Deb, ce qui fait de cet algorithme un des plus utilisés aujourd'hui.

Fonctionnement général

NSGA-II est un algorithme élitiste n'utilisant pas d'archive externe pour stocker l'élite. Pour gérer l'élitisme, *NSGA-II* assure qu'à chaque nouvelle génération, les meilleurs individus rencontrés soient conservés.

Pour comprendre le fonctionnement de l'algorithme, plaçons nous à la génération t. Comme le montre la Figure 2.5, deux populations (P_t et Q_t de taille N) coexistent. La population P_t contient les meilleurs individus rencontrés jusqu'à la génération t, et la population Q_t est formée d'individus autres, issus des phases précédentes de l'algorithme. La première étape consiste à créer la population $R_t = P_t \bigcup Q_t$ et à appliquer une procédure de *ranking* (cf. Section 2.3.2) pour identifier les différents fronts F_i de solutions non dominées (les meilleurs individus se retrouvent donc dans le, ou les, tous premiers fronts).

La deuxième phase consiste à construire une nouvelle population P_{t+1} contenant les N meilleurs individus de R_t. Il faut pour cela inclure intégralement les meilleurs fronts F_i (c'est-à-dire en commençant à l'indice 1) tant que le nombre d'individus présents dans P_{t+1} est inférieur à N. Il reste donc à ce stade $N - |P_{t+1}|$ individus à inclure dans P_{t+1}. Pour cela, une procédure de *crowding* est appliquée sur le premier front F_i non inclus. Les $N - |P_{t+1}|$ meilleurs individus au sens de cette procédure de crowding sont insérés dans P_{t+1}.

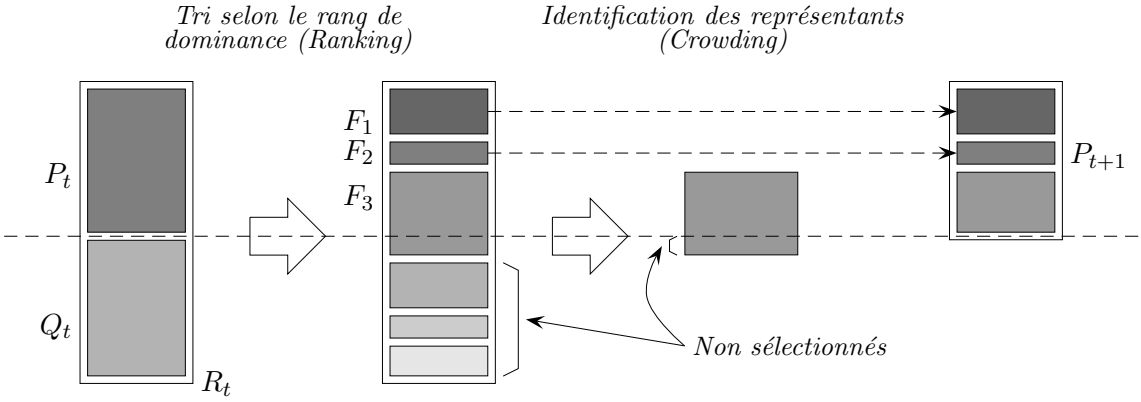


FIG. 2.5 – *Fonctionnement de* `NSGA-II` *: schéma extrait de [Deb, 2001].*

La troisième phase consiste à remplir la population Q_{t+1}. Il suffit alors d'utiliser les opérateurs de sélection, croisement et mutation sur les individus de P_{t+1}, puis d'insérer les descendants dans Q_{t+1}. L'algorithme 3 reprend toutes les étapes décrites ci-dessus.

Algorithme 3 Pseudo-code de l'algorithme *`NSGA-II`*.

Initialiser les populations P_0 et Q_0 de taille N
`Tant que` *critère_d'arrêt_non_rencontré* ***`faire`***
- Création de $R_t = P_t \bigcup Q_t$
- Calcul des différents fronts F_i de la population R_t par un algorithme de "ranking"
- Mettre $P_{t+1} = \emptyset$ et $i = 0$,
- `Tant que` $|P_{t+1}| + |F_i| < N$ `faire`
  $P_{t+1} = P_t \bigcup F_i$
  $i = i + 1$
  `FinTantQue`
- Inclure dans P_{t+1} les $(N - |P_{t+1}|)$ individus de F_i les mieux répartis au sens de la distance de "crowding"
- Sélection dans P_{t+1} et création de Q_{t+1} par application des opérateurs de croisement et mutation

`FinTantQue`

Calcul de la distance de "crowding"

Le calcul de valeur d'adaptation pour *`NSGA-II`* décrit dans [Deb *et al.*, 2000] ne sert pas uniquement pour la sélection des opérateurs de croisement et de mutation, mais intervient aussi dans la sélection des individus à inclure dans P_{t+1} (la population contenant les élites). C'est donc une phase importante pour laquelle les auteurs de *`NSGA-II`* ont développé une

méthode particulière : *la distance de crowding.*

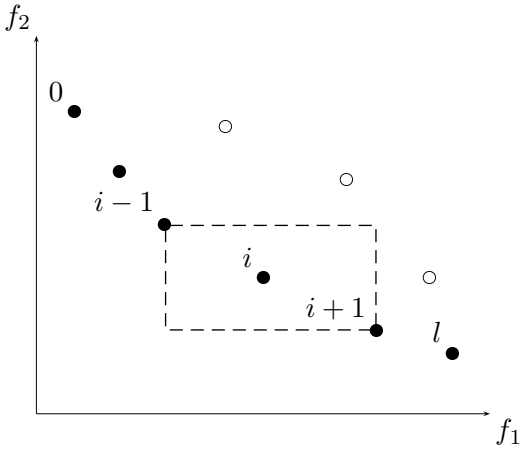


FIG. 2.6 – *Calcul d'un représentant (crowding) [Deb, 2001].*

La *distance de crowding* d_i d'un point particulier i se calcule en fonction du périmètre de l'hypercube ayant comme sommets les points les plus proches de i sur chaque objectif. Sur la figure 2.6 est représenté l'hypercube en deux dimensions associé au point i. Un algorithme de calcul de la distance de crowding est détaillé dans [Deb, 2001]. Cet algorithme est de complexité $\mathcal{O}(MN\log(N))$, où M est le nombre d'objectifs du problème et N le nombre d'individus à traiter. Une fois tous les d_i calculés, il ne reste plus qu'à les trier par ordre décroissant et à sélectionner les individus possédant la plus grande valeur de crowding.

Les techniques de sélection en vue d'un croisement ou d'une mutation peuvent aussi bénéficier de ce type de calcul. En effet, l'opérateur de sélection le plus fidèle à l'esprit de Pareto tombe en défaut dès que les deux individus en compétition ont le même rang de dominance (i.e. sont non dominés). Pour pallier à ce problème, `NSGA-II` utilise la distance de "crowding" pour départager les deux individus. Ce processus nommé *Crowded Tournament Selection* permet de conserver les bonnes propriétés de la sélection Pareto tout en préservant une certaine diversité sur les individus sélectionnés.

2.4.2 Le "Strength Pareto Evolutionary Algorithm" (`SPEA`)

La méthode `SPEA` implémentée par Zitzler et Thiele [Zitzler and Thiele, 1998b] est l'illustration même d'un algorithme évolutionnaire élitiste. Pour réaliser cet élitisme, `SPEA` maintient une archive externe contenant le meilleur front de compromis rencontré durant la recherche.

Fonctionnement général

La première étape consiste à créer une population initiale (constituée par exemple d'individus générés aléatoirement). L'archive externe, au départ initialisée à l'ensemble vide, est mise à jour régulièrement en fonction des individus non dominés de la population.

À chaque itération de l'algorithme, on retrouve les étapes classiques *sélection + croisement + mutation*. Puis les nouveaux individus non dominés découverts viennent s'ajouter à l'archive, et les individus de l'archive dominés par le nouvel arrivant sont supprimés. Si l'archive vient à excéder une certaine taille (fixée au départ), alors une phase de clustering est appliquée dans le but de garder les meilleurs représentants.

La figure 2.7 (extraite de la thèse de Zitzler [Zitzler, 1999]) et l'algorithme 4 illustrent le schéma général de fonctionnement de l'algorithme.

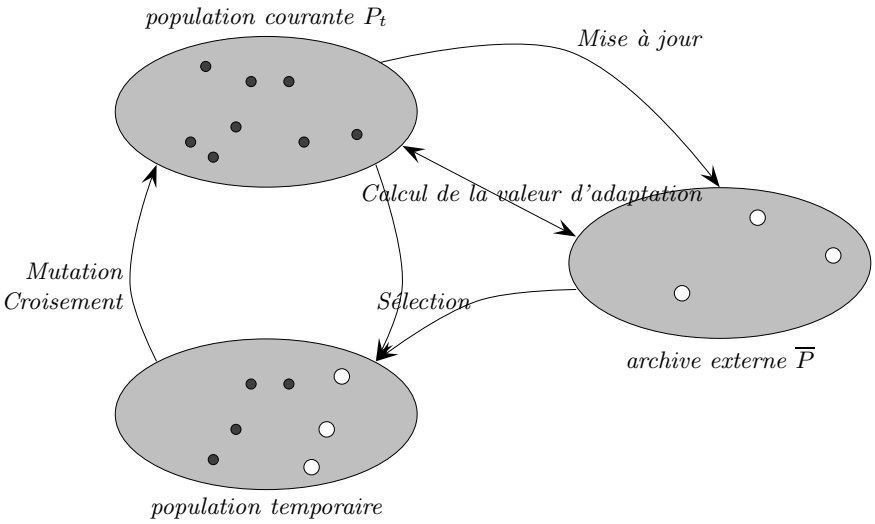


FIG. 2.7 – *Fonctionnement général de l'algorithme* `SPEA`.

Algorithme 4 Pseudo-code de l'algorithme *`SPEA`*.

Initialiser la population P_0 et créer l'archive externe vide $\overline{P} = \emptyset$
Mise à jour de $\overline{P}$ à partir des individus non dominés de P_0
`Tant que` *critère_d'arrêt_non_rencontré* `faire`
- Calcul de la valeur d'adaptation pour tous les individus de $P + \overline{P}$
- Sélection dans $P_t + \overline{P}$ en fonction de la valeur d'adaptation
- Croisement
- Mutation
- Mise à jour de $\overline{P}$ à partir des individus non dominés de P

`FinTantQue`

Calcul de la valeur d'adaptation

Le calcul de la valeur d'adaptation de *SPEA* permet de diminuer les chances de sélection des individus ayant une moins bonne évaluation, ou des individus issus de niches déjà importantes. À la différence des techniques classiques de "sharing" qui sont basées sur la notion de distance euclidienne dans l'espace des objectifs, *SPEA* utilise la notion de dominance pour détecter les niches d'individus. De plus, de par la nature élitiste de l'algorithme, les individus de l'archive externe participent eux aussi à la sélection, le calcul d'adaptation doit donc les prendre en compte.

Ce calcul s'effectue en deux étapes. La première étape consiste à affecter une valeur, appelée valeur de dureté (S), aux individus de l'archive $\overline{P}$ externe. Cette valeur est déduite à partir du nombre d'individus qu'un élément $i \in \overline{P}$ domine faiblement dans la population courante P_t :

$$S(i) = \frac{|\{j \mid j \in P_t \wedge f(i) \leq f(j)\}|}{|P_t| + 1}$$

La valeur d'adaptation d'un individu $i \in \overline{P}$ est égale à sa valeur de dureté : $F(i) = S(i)$. Pour un individu j de la population courante P_t, la valeur d'adaptation est calculée en réalisant la somme des valeurs de dureté des individus de $\overline{P}$ dominant j :

$$F(j) = 1 + \sum_{i \in \overline{P} \wedge f(i) \leq f(j)} S(i)$$

Le chiffre 1 est ajouté au total de la somme pour éviter que des individus de $\overline{P}$ aient une valeur d'adaptation plus grande que certains individus de P_t. Il est à noter que, dans ce calcul, une plus petite valeur d'adaptation conduit à une plus grande chance de sélection de l'individu. Ainsi, plus un individu est dominé par les individus de $\overline{P}$ et plus sa valeur d'adaptation décroît, diminuant donc ses chances d'être sélectionné. Un exemple de calcul de valeurs d'adaptation est illustré à la figure 2.8. En observant cette figure, il est clair que l'individu évalué à 19/6 (qui est le résultat de la somme : $1 + 3/6 + 2/6 + 3/6 + 3/6 + 2/6$) à une probabilité moindre d'être sélectionné que les individus évalués à 14/6..

Réduction par clustering

Lorsque le nombre d'individus de l'archive externe est grand, les performances de l'algorithme peuvent se dégrader significativement. En effet, le nombre d'individus influe sur le calcul de la valeur d'adaptation, celui-ci devient moins fiable, et peut tromper la recherche en la focalisant, par exemple, sur des zones déjà explorées.

Pour pallier à ce défaut, une solution consiste à utiliser des techniques de "clustering". Ces techniques ont été étudiées intensivement dans le contexte des *analyses de cluster* [Morse, 1980] et ont été appliquées avec succès pour déterminer des partitions d'une collection relativement hétérogène d'éléments. La technique de "clustering" utilisée par *SPEA* est assez intuitive. Au début de la procédure, chaque individu constitue son propre groupe, puis on fusionne deux à deux les groupes les plus proches en terme de distance. Cette étape est itérée jusqu'à l'obtention du nombre désiré de groupes. Une fois les groupes

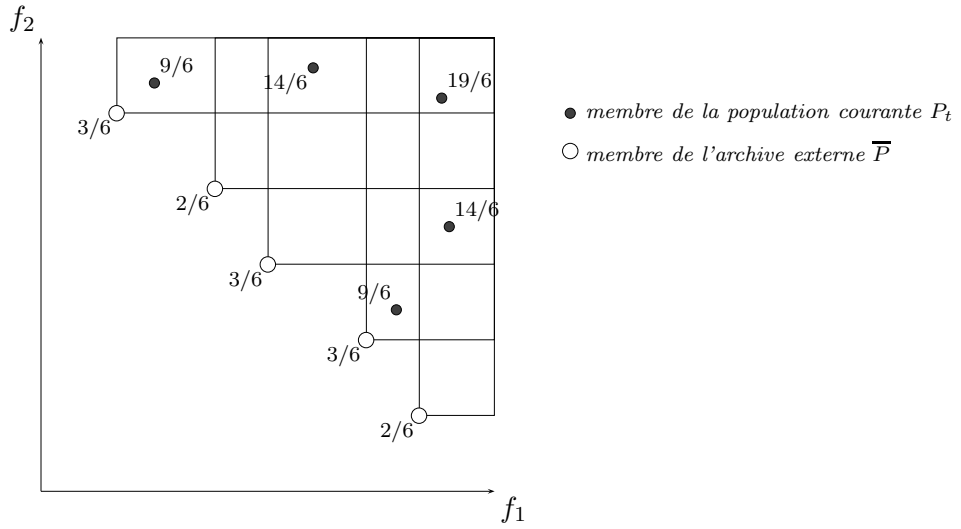


FIG. 2.8 – *Calcul des valeurs d'adaptation en dimension* 2.

identifiés, il ne reste plus qu'à choisir un représentant par groupe. Ce représentant peut être déterminé de plusieurs façons, par exemple en prenant le barycentre du groupe. C'est ce représentant qui sera gardé, les autres éléments étant tout simplement supprimés. Les étapes de la méthode de clustering sont illustrées à la figure 2.9.

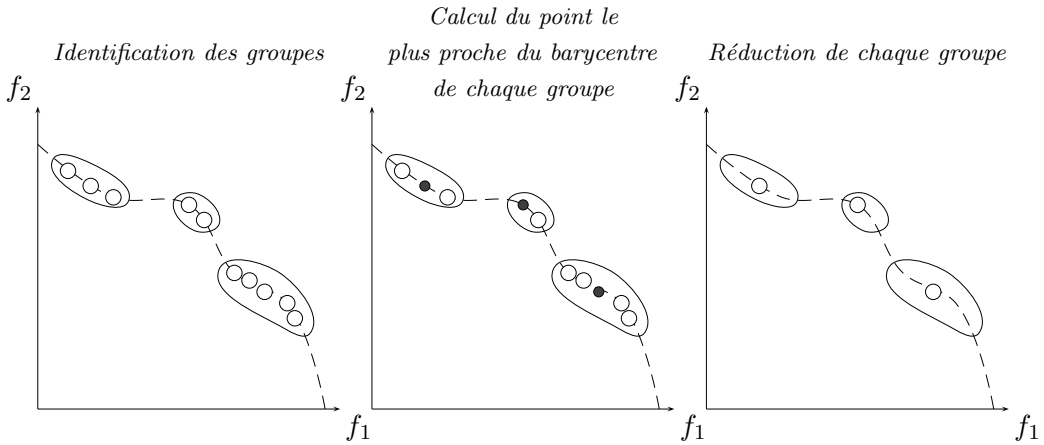


FIG. 2.9 – *Illustration du clustering en dimension* 2.

2.5 Discussion

Nous avons présenté dans ce chapitre quelques méthodes Pareto issues des métaheuristiques pour l'optimisation multiobjectif. Ces méthodes ont montré leur efficacité pour trouver des solutions approchées satisfaisantes pour un grand nombre de problèmes.

Cependant, ces méthodes ont aussi leurs défauts, le plus gros d'entre eux concernant l'impossibilité de juger la qualité de l'approximation par rapport au front Pareto optimal.

En effet, il est généralement impossible d'estimer l'erreur séparant la solution approchée fournie du front Pareto optimal du problème.

Dans cette thèse, nous allons proposer dans les chapitres 6 et 7 une méthode hybride permettant à la fois de calculer des solutions approchées et de borner le front Pareto.

Chapitre 3

La mesure des surfaces de compromis

Dans ce chapitre, nous présentons quelques mesures permettant d'évaluer un aspect de la surface de compromis produite par un algorithme. L'évaluation des surfaces de compromis est un problème délicat qui oblige à utiliser plusieurs mesures différentes, car il est impossible de représenter par une unique valeur réelle la qualité des solutions, la taille du front, la répartition des solutions sur le front, ...

Sommaire

3.1 Introduction

Les différentes méthodes de résolution présentées au chapitre 2 tentent de calculer la meilleure approximation du front Pareto. Or, ces méthodes intégrant un aspect stochastique, plusieurs exécutions ne donnent pas toujours les mêmes résultats. De plus, différentes méthodes donnent aussi des résultats différents. Il est donc nécessaire de pouvoir comparer les différents résultats calculés lors de la phase d'optimisation, dans le but de comparer l'efficacité des méthodes.

Comment comparer deux ensembles de compromis ? La réponse naturelle est de considérer l'ensemble lui-même, mais il n'est pas facile de déterminer de manière automatique quel est le meilleur ensemble. En effet, plusieurs aspects entrent en ligne de compte, notamment : la qualité (la proximité par rapport au front Pareto théorique), la distribution (est-ce que toutes les parties du front Pareto sont découvertes), la répartition (est-ce que les points sont répartis de manière homogène sur le front), ... Il est très difficile, voire impossible, de prendre en compte tous ces paramètres au travers d'une seule valeur numérique. C'est pourquoi il est courant d'utiliser plusieurs mesures (ou métriques) pour tester tel ou tel aspect de l'ensemble.

De plus, il faut distinguer deux types de métriques : les métriques relatives, qui comparent deux ensembles, et les métriques absolues, qui évaluent un ensemble sans avoir besoin d'autres points ou ensemble de référence.

Dans la suite de ce chapitre, nous allons présenter quelques mesures couramment utilisées. Nous mettrons en avant leurs points forts et leurs points faibles, de manière à aiguiller le choix du développeur quant à la métrique à employer.

3.2 La métrique d'espacement

La première métrique présentée ici est la métrique d'espacement (*spacing metric*), introduite par Schott [Schott, 1995]. Celle-ci permet de mesurer l'uniformité de la répartition des points composant la surface de compromis. La définition de cette métrique est la suivante [1] :

Définition 17 La métrique d'espacement : *Soit A un ensemble de n éléments de dimension m, alors $\mathcal{S}(A)$ est définie par :*

$$\mathcal{S}(A) = \left[\frac{1}{n-1} \times \sum_{i=1}^{n} (\overline{d} - d_i)^2 \right]^{\frac{1}{2}}$$

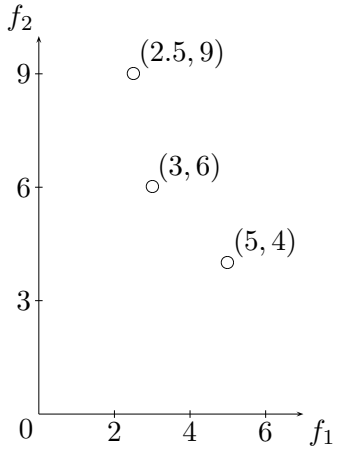
avec

$d_i = \min_j \left(|f_1^i(\overrightarrow{x}) - f_1^j(\overrightarrow{x})| + \cdots + |f_m^i(\overrightarrow{x}) - f_m^j(\overrightarrow{x})| \right) \qquad i, j \in [1, \cdots, n]$ *et* $i \neq j$
$\overline{d}$ *: moyenne de tous les* d_i

1. La définition présentée est tirée de l'ouvrage [Collette and Siarry, 2002]

Cette métrique permet de mesurer l'uniformité de la répartition des points de la surface de compromis dans l'espace $(f_1,\ldots,f_m)$. Comme nous l'avons évoqué au chapitre 2, la diversité est un atout important pour une méthode d'optimisation. Une bonne diversité influera fortement sur la répartition des points sur la surface de compromis.

Exemple : Nous reprenons ici l'exemple donné dans [Collette and Siarry, 2002] pour illustrer la métrique d'espacement. Cet exemple considère une surface de compromis composée de trois points de $\mathbb{R}^2$. La surface de compromis, ainsi que le tableau des distances, sont représentés à la Figure 3.1.



$\Longrightarrow$

Point i	Point j	Distance
$(2.5, 9)$	$(3, 6)$	3.5
$(2.5, 9)$	$(5, 4)$	7.5
$(3, 6)$	$(5, 4)$	4

FIG. 3.1 – *Exemple de calcul de la métrique d'espacement.*

On obtient donc les valeurs d_i suivantes :

Pour le point de coordonnées $(2.5, 9)$, $d_1 = 3.5$.

Pour le point de coordonnées $(3, 6)$, $d_2 = 3.5$.

Pour le point de coordonnées $(5, 4)$, $d_3 = 4$.

On peut maintenant calculer la moyenne des d_i : $\overline{d} = \frac{3.5+3.5+4}{3} = 3.67$.

D'où : $\mathcal{S}(A) = \left[\frac{1}{2} \times ((3.67 - 3.5)^2 + (3.67 - 3.5)^2 + (3.67 - 4)^2)\right]^{\frac{1}{2}}$

Soit $\mathcal{S}(A) = 0.288$

Plus le résultat de la mesure est proche de 0, meilleure est la répartition des points sur la surface de compromis. Cette métrique est un moyen de mettre en avant l'aspect "diversité" d'un algorithme, mais, dans certains cas, l'évaluation peut être biaisée. En effet, si le front Pareto est discontinu, la valeur de $\mathcal{S}$ est trop élevée. Les trous présents dans la surface de compromis influent sur le calcul de $\mathcal{S}$ et donc l'interprétation du résultat est difficile. Il est important de vérifier ces aspects avant d'utiliser la métrique d'espacement.

3.3 L'hypervolume

Cette métrique introduite par Zitzler pendant sa thèse, calcule une approximation du volume compris sous la courbe formée par les points de l'ensemble à évaluer. Ainsi, lorsque le problème comporte deux critères, ce calcul correspond au calcul d'une aire (cf. figure 3.2) , lorsque le problème comporte trois critères, la valeur calculée est un volume .... La définition présentée dans [Zitzler, 1999] est la suivante :

Définition 18 L'hypervolume : *Soit* $A = (x_1, x_2, ..., x_n) \subseteq X$ *un sous-ensemble de* n *éléments. La fonction* $\mathcal{H}$ *calcule le volume borné par l'union de tous les polytopes* $p_1, p_2, ..., p_n$*, où chaque* p_i *est formé par l'intersection des hyperplans des* x_i *par rapport aux axes du repère : pour chaque axe de l'espace des objectifs, il existe un hyperplan perpendiculaire à l'axe et passant par le point* $(f_1(x_i), f_2(x_i), ..., f_k(x_i))$*. Pour le cas à deux dimensions, chaque* p_i *représente un rectangle défini par les points de coordonnées* $(0, 0)$ *et* $(f_1(x_i), f_2(x_i))$*.*

Dans le cas de problèmes de minimisation où l'on cherche à minimiser les valeurs des objectifs, et lorsque les surfaces de compromis comportent le même nombre de points, plus la valeur d'hypervolume est petite, meilleure sera la surface de compromis.

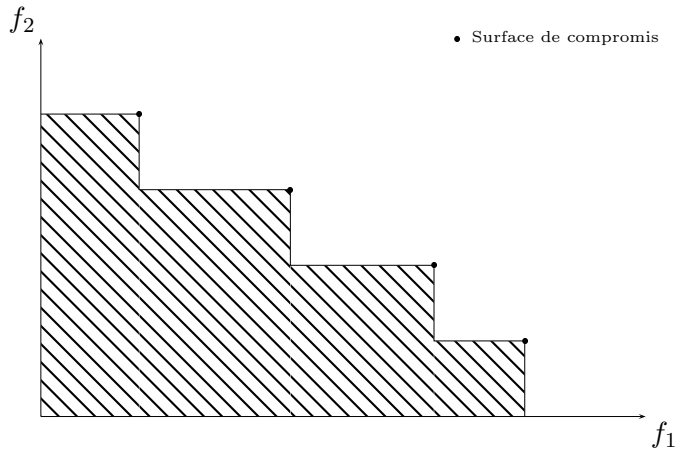


FIG. 3.2 – *Calcul d'hypervolume d'une surface de compromis.*

La mesure $\mathcal{H}$ permet de donner une valeur numérique simple à une surface de compromis composée a priori de plusieurs points. Il est clair que la valeur calculée ne permet de mesurer qu'un aspect de la surface de compromis et que, dans certains cas, comme les problèmes non convexes (cf. Veldhuizen [Veldhuizen, 1999]), la valeur calculée peut même être erronée.

Un des principaux avantages de cette métrique réside dans le fait qu'elle ne nécessite aucun autre point ou ensemble de référence. Ainsi, elle permet d'évaluer les performances d'une méthode de manière complètement indépendante.

Un des principaux reproches qui lui était attribué jusqu'ici était son fort coût calculatoire. En effet, pour calculer l'union des hypervolumes induits par chaque point de la

surface de compromis, il est nécessaire de calculer un grand nombre d'intersections de volumes. Cette opération, très rapide dans le cas d'un problème à deux objectifs, devient coûteuse lorsque le nombre de critères dépasse 3. Le temps de calcul, dû à une complexité polynomiale élevée, est si important qu'il excède le temps de résolution du problème traité par un algorithme génétique. Très récemment, Fleischer dans [Fleischer, 2003] a proposé une méthode efficace, de complexité faible pour le calcul de cet hypervolume. Ce résultat permet non seulement le calcul rapide d'hypervolumes lors de la comparaison d'algorithmes, mais l'auteur a aussi réussi à combiner cet opérateur avec une métaheuristique (le recuit simulé). La méthode d'optimisation ainsi développée résout les problèmes en se servant de l'hypervolume comme d'une fonction objectif.

Forte de ces avancées récentes, la métrique de l'hypervolume est importante à intégrer dans l'échantillon des outils de mesure nécessaires pour l'évaluation des méthodes d'optimisation.

3.4 La métrique C

Comme l'hypervolume, cette métrique a été introduite par Zitzler [Zitzler, 1999]. C'est une métrique relative, qui compare deux surfaces de compromis A et B. La valeur $\mathcal{C}(A,B)$ correspond au pourcentage d'éléments de B faiblement dominés par au moins un des éléments de A. Le calcul de ce ratio s'effectue grâce à la formule suivante :

Définition 19 La couverture de deux ensembles : *Soient $A, B \subseteq X$ deux ensembles de vecteurs de décisions. La fonction $\mathcal{C}$ associe à chaque paire ordonnée (A,B) une valeur de l'intervalle $[0,1]$:*

$$\mathcal{C}(A,B) : = \frac{|\{b \in B \mid \exists\, a \in A : a \geq b\}|}{|B|}$$

La valeur $\mathcal{C}(A,B) = 1$ signifie que tous les vecteurs de décisions de B sont faiblement dominés[2] *par ceux de A. À l'inverse, $\mathcal{C}(A,B) = 0$, signifie qu'aucun des points de B n'est faiblement dominé par un point de A.*

Ainsi, plus la valeur $\mathcal{C}(A,B)$ se rapproche de 1, meilleure la surface de compromis A est considérée par rapport à B. La métrique $\mathcal{C}$ n'étant pas symétrique : $\mathcal{C}(A,B) \neq 1 - \mathcal{C}(B,A)$, il est nécessaire de considérer les deux valeurs $\mathcal{C}(A,B)$ et $\mathcal{C}(B,A)$ pour obtenir une mesure plus fiable des deux surfaces de compromis A et B à comparer.

Toutefois, dans certains cas particuliers, des problèmes peuvent apparaître. Considérons le cas présenté à la figure 3.3, le problème considéré étant un problème de minimisation. Il apparaît clairement que le front B est une meilleure approximation du front Pareto que le front A. Cependant la valeur de $\mathcal{C}(A,B)$ sera la même que celle de $\mathcal{C}(B,A)$: $\mathcal{C}(A,B) = \mathcal{C}(B,A) = \frac{1}{2}$. Dans ce cas de figure, la métrique $\mathcal{C}$ n'est pas en mesure de différencier les deux surfaces A et B.

L'algorithme de calcul de la métrique C est de complexité polynomiale en $\mathcal{O}(n^2)$. Ce coût raisonnable ne doit pas faire oublier le fait qu'il faut considérer deux calculs de C, la mesure n'étant pas symétrique.

2. *Un point u "domine faiblement" un point v si et seulement si, $\forall i \in \{1,2,...,k\} : u_i \geq v_i$.*

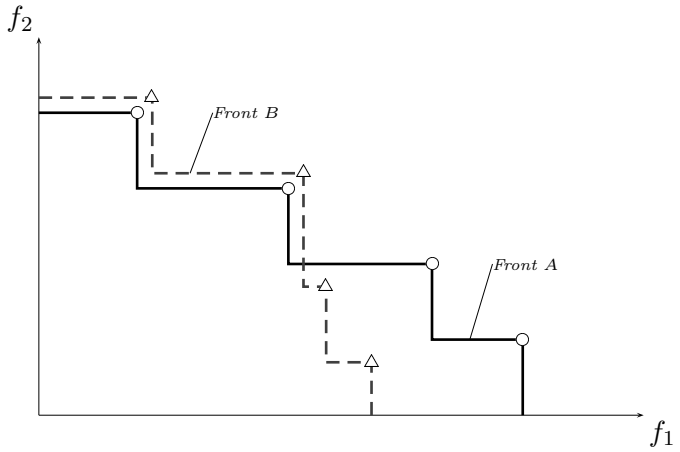


FIG. 3.3 – *Problème lors de l'utilisation de la métrique $\mathcal{C}$.*

La métrique C est un outil intéressant de mesure relative. Elle permet de différencier nettement deux surfaces de compromis lorsque d'autres mesures (l'hypervolume par exemple) donnent des évaluations trop proches. Elle doit cependant être utilisée conjointement à d'autres pour ne pas fausser le jugement, notamment lors de cas tels que celui présenté figure 3.3.

3.5 Discussion

Nous avons présenté dans ce chapitre quelques métriques parmi les plus utilisées aujourd'hui. Chacune de ces métriques mesure un aspect (forme, uniformité, qualité, ...) de la surface de compromis, parfois de manière absolue, et d'autres fois en comparaison avec une autre surface.

Au travers des surfaces de compromis, c'est souvent l'algorithme qui est évalué. Les métriques relatives sont donc souvent utilisées pour comparer deux algorithmes, et essayer de les départager sur l'instance de problème considérée.

Il est important de retenir qu'aucune des mesures existantes ne peut synthétiser en une seule valeur toute l'information contenue dans la surface de compromis. C'est pourquoi il est souvent nécessaire d'utiliser plusieurs de ces mesures conjointement pour espérer évaluer au mieux la surface de compromis. Et, même avec plusieurs métriques, il y aura encore des aspects qui ne sont pas pris en compte. Le meilleur moyen d'évaluer une surface de compromis, lorsque le problème ne comporte pas plus de trois critères, reste encore l'évaluation graphique. C'est pourquoi la représentation graphique des surfaces de compromis (sur les problèmes le permettant) est considérée comme une information importante lors de l'évaluation des résultats.

Deuxième partie

Le Sac à Dos multidimensionnel multiobjectif

Chapitre 4

Une étude empirique de la recherche Tabou pour le problème du sac à dos multidimensionnel multiobjectif

Dans ce chapitre, nous montrons que les algorithmes de recherche locale sans utilisation de population permettent de calculer une bonne approximation de la surface de compromis. Nous démontrons expérimentalement que la clef pour obtenir une telle efficacité est liée à l'incorporation dans la méthode d'un mécanisme "intelligent" de diversification. Pour illustrer nos propos, nous présentons trois algorithmes de recherche Tabou intégrant des mécanismes de diversification différents. Nous montrons ainsi que la diversité est un facteur important influant sur les résultats des algorithmes.

Sommaire

4.1 Le sac à dos multidimensionnel multiobjectif (MOKP)

Dans ce chapitre et le suivant, nous allons nous intéresser à la résolution de problèmes combinatoires multiobjectifs, c'est-à-dire de problèmes dont les variables sont discrètes. Il existe beaucoup de problèmes combinatoires réputés. Parmi eux le problème du sac à dos multidimensionnel multiobjectif est l'un des plus étudiés dans la communauté multiobjectif. Nous avons donc choisi ce problème pour nos travaux et nous allons maintenant le présenter.

Soit un ensemble d'items (ou objets), à chacun d'entre eux sont associés des valeurs de profit ainsi que des poids. Le problème du sac à dos multidimensionnel multiobjectif en 0-1 (MOKP01) consiste à sélectionner un sous-ensemble d'objets maximisant une fonction multiobjectif, exprimée en fonction des valeurs de profit, tout en respectant un ensemble de contraintes de type sac à dos. Le problème du MOKP01 peut être défini plus formellement de la manière suivante :

$$\text{MOKP01} \begin{cases} \max & z^j(x) = \sum_{i=1}^{n} c_i^j x_i & j = 1, ..., m \\ \text{t.q.} & \sum_{i=1}^{n} w_i^l x_i \leq b_l & l = 1, ..., q \\ & x_i \in \{0, 1\} & i = 1, ..., n \end{cases}$$

où n est le nombre d'objets, x_i une variable de décision, m le nombre d'objectifs, z^j la $j^{ème}$ composante de la fonction multiobjectif z et q le nombre de contraintes sac à dos du problème.

Comme le problème NP-difficile du sac à dos multidimensionnel (MKP), le MOKP peut être utilisé pour modéliser de nombreux problèmes réels, comme la répartition de budgets et l'allocation de ressources. Plusieurs algorithmes heuristiques ont été développés pour résoudre le MOKP. Des méthodes de voisinage comme le recuit simulé [Serafini, 1992; Ulungu *et al.*, 1999] et la recherche Tabou [Gandibleux *et al.*, 1997], des méthodes évolutionnaires : "vector evaluated genetic algorithm" (VEGA) [Schaffer, 1984], "non-dominated sorting genetic algorithm" (NSGA) [Srinivas and Deb, 1994], "strength Pareto evolutionary algorithm" (SPEA) [Zitzler and Thiele, 1998a], et "memetic Pareto archived evolution strategy algorithm" (M-PAES) [Corne and Knowles, 2000]. Très récemment, des algorithmes hybrides combinant la recherche génétique et la recherche locale (limitée à des méthodes de descente à notre connaissance) tels que "multiple objective genetic local search" (MOGLS) [Jaszkiewicz, 2002], ainsi que des approches hybrides parallèles, [Jozefowiez *et al.*, 2002] ont été proposés.

Les travaux précédents tendent à mettre en avant les avantages des algorithmes évolutionnaires (AE), utilisant une population d'individus, sur les algorithmes de recherche locale dans le contexte de l'optimisation multiobjectif. En effet, étant donné que résoudre un problème multiobjectif passe par le calcul d'un *ensemble* de solutions *compromis* (aussi près que possible du front Pareto), le concept de population utilisé par les AEs est naturellement bien adapté. Dans une telle situation, il est clair que la diversité des solutions

joue un rôle important dans l'efficacité de la recherche.

Nous démontrons expérimentalement que la clef pour obtenir une telle efficacité est liée à l'incorporation dans la méthode d'un mécanisme "intelligent" de diversification. Pour illustrer nos propos, nous présentons trois algorithmes de recherche Tabou pour le MOKP. Le premier algorithme (appelé *TS*) est un algorithme Tabou classique où la diversification n'est assurée que par la liste Tabou. Le second algorithme (*TS+RW*) intègre une procédure de marche aléatoire pour diversifier la recherche. Le dernier algorithme (*TS+HW*) utilise un procédé original de mesure de la diversité au cours de la recherche, lui permettant soit de continuer dans la même direction de recherche (intensification), soit de déclencher des phases de diversification pour réorienter la recherche.

Pour montrer l'influence de ces techniques de diversification, nous avons effectué des expérimentations utilisant des instances connues du MOKP. Nous comparons notre meilleur algorithme Tabou avec deux algorithmes évolutionnaires réputés (*SPEA* et *MOGLS*). Nous montrons ainsi que, bien que notre algorithme Tabou n'opère qu'avec une seule solution durant la recherche, il rivalise avec les algorithmes évolutionnaires utilisant une population.

4.2 Les algorithmes Tabou

Pour que la recherche Tabou, présentée à la Section 2.2.4, soit performante, elle doit établir un bon compromis entre *exploration* et *exploitation* durant la recherche. Ainsi, l'exploration et l'exploitation sont respectivement assurées par des opérateurs de *diversification* et *d'intensification*. Nous mettons en avant, dans les sections suivantes, l'importance de la diversification dans le cadre des problèmes d'optimisation multiobjectifs.

4.2.1 Les caractéristiques communes de nos algorithmes Tabou

Nous présentons maintenant les caractéristiques communes de nos trois algorithmes Tabou.

L'espace de recherche.

Une configuration est donnée par un vecteur binaire $x = (x_1, x_2, ..., x_n)$ vérifiant toutes les contraintes de sac à dos, c.à.d, $\forall l \in \{1, ..., q\}, \sum_{i=1}^{n} w_i^l x_i \leq b_l$. L'espace de recherche S est alors composé de tous ces vecteurs binaires, c.à.d, $S = \{x \in \{0,1\}^n | \sum_{i=1}^{n} w_i^l x_i \leq b_l, l = 1, ..., q\}$.

Le voisinage.

Le voisinage N de notre problème est défini de la manière suivante : soient $x, x' \in \mathcal{S}$, x et x' sont voisins ($x' \in N(x)$) s'ils diffèrent par la valeur d'une seule variable. Plus formellement, $N(x) = \{x' \in S \mid distanceHamming(x, x') = 1\}$. Il en résulte qu'à partir d'une configuration x, il est possible d'obtenir une configuration voisine x' en ajoutant ou en enlevant un objet ($x_i : 0 \rightarrow 1$ ou $1 \rightarrow 0$) de x tel que x' reste réalisable. Le mouvement

de x vers x' est alors caractérisé par l'entier i représentant l'indice de la variable x_i qui a été changée. C'est cet indice qui sera considéré comme l'attribut du mouvement.

Évaluation du voisinage.

L'évaluation du voisinage est basée sur l'approche pondérée de Tchebycheff [Miettinen, 1999; Ehrgott and Gandibleux, 2000; Deb, 2001] [1]. Cette approche peut se décrire formellement ainsi : soient $x = (x_1, x_2, ..., x_n) \in S$ et $x' = (x'_1, x'_2, ..., x'_n) \in N(x)$, l'évaluation de x' est définie par la fonction suivante :

$$eval(x, x') = \min_j \lambda_j (z^j(x') - z^j(x))$$

où

- $z^j(x') = \sum_{i=1}^{n} c_i^j \times x'_i$, est la valeur du $j^{ème}$ objectif de x' ;
- λ_j, est la $j^{ème}$ composante du vecteur $\lambda \in [0,1]^m$, où $(\sum_{j=1}^{m} \lambda_j) = 1$. Le vecteur λ correspond à une direction dans l'espace des objectifs, et permet d'orienter la recherche dans une autre direction, donc de la diversifier.

Choix du meilleur voisin.

Soit x la configuration courante de l'algorithme. La configuration voisine x' devant remplacer x est déterminée grâce à l'équation suivante : $eval(x, x') = \max_{y \in N(x)} eval(x, y)$. Si plusieurs configurations vérifient l'équation, l'algorithme en choisit une aléatoirement.

Gestion de la liste Tabou.

Chaque fois qu'un mouvement est appliqué pour aller de x à x', l'indice de la variable changée est enregistré dans la liste Tabou. Ainsi, le mouvement inverse (correspondant au retour à la configuration de départ) est interdit pour les k prochaines itérations de l'algorithme. Pour implémenter la liste Tabou, nous utilisons un tableau T de n éléments. Chaque fois qu'un indice est ajouté à la liste, la valeur du nombre courant d'itérations augmenté de la valeur Tabou k est placée dans la case de T correspondant à l'élément ajouté. À chaque instant, il est facile de vérifier si un mouvement donné est marqué comme Tabou ou non, simplement en comparant le nombre courant d'itérations avec celui enregistré dans le tableau T.

Le critère d'aspiration.

Le mécanisme précédent est suffisant pour empêcher l'algorithme de rester bloqué dans des cycles courts. Cependant, un tel mécanisme peut interdire certaines configurations qui n'ont pas été encore visitées. Pour remédier à ce problème, un critère d'aspiration standard est introduit. Un mouvement marqué comme Tabou est quand même choisi si

1. Deux autres approches réputées sont basées sur des techniques d'agrégation [Cohon, 2000] ou de *ranking* [Goldberg, 1989].

celui-ci conduit à l'obtention d'une configuration dont l'évaluation est meilleure que la meilleure configuration rencontrée jusqu'ici par l'algorithme.

4.2.2 Caractéristiques spécifiques des trois algorithmes Tabou

Nous introduisons maintenant nos trois algorithmes Tabou : l'algorithme Tabou standard (*TS*), l'algorithme Tabou avec marche aléatoire (*TS+RW*) et l'algorithme Tabou utilisant une mesure de la distance de Hamming (*TS+HW*).

L'algorithme Tabou standard *TS*.

L'algorithme Tabou standard *TS* résulte directement du schéma présenté à la Section 4.2.1 (cf. algorithme 5). La liste Tabou assure seule le rôle de la diversification.

Algorithme 5 L'algorithme Tabou standard *TS*.

Soit x une configuration réalisable, et L le nombre d'itérations
$ND \leftarrow \{x\}$ *(ND est l'ensemble des solutions non dominées)*
`Pour` $i = 0$ `jusqu'à` L `faire`
    Choisir le meilleur mouvement m autorisé
    Mettre à jour la liste Tabou en fonction de m
    Effectuer le mouvement m dans x
    Mettre à jour l'ensemble des solutions non dominées ND en fonction de x
`FinPour`
Renvoyer ND

L'algorithme Tabou avec marche aléatoire *TS+RW*.

Le principe de la marche aléatoire (*Random Walk (RW)*) est bien connu dans la littérature et couramment utilisé dans des algorithmes réputés comme *WSAT* [Selman *et al.*, 1994]. La marche aléatoire consiste à réaliser de temps en temps un mouvement qui n'est plus guidé par la fonction d'évaluation et constitue donc un schéma de diversification. Intégrer la marche aléatoire dans un algorithme Tabou standard permet d'obtenir une recherche plus diversifiée.

À chaque itération de l'algorithme *TS+RW*, une valeur réelle $rw \in [0, 1]$ est choisie aléatoirement. Soit $rw_{seuil} \in [0, 1]$ la valeur seuil, alors, si $rw > rw_{seuil}$ l'algorithme exécutera un mouvement classique (cf. Section 4.2.1), dans le cas contraire, l'algorithme effectuera un mouvement aléatoire réalisable (cf. Algorithme 6). L'algorithme *TS+RW* peut être décrit ainsi :

Algorithme 6 L'algorithme Tabou avec marche aléatoire *TS+RW*.

Soit x une configuration réalisable, L le nombre d'itérations,
et rw_{seuil} le seuil de perturbation
$ND \leftarrow \{x\}$ *(ND est l'ensemble des solutions non dominées)*
`Pour` $i = 0$ `jusqu'à` L `faire`
    Tirer aléatoirement un nombre $rw \in [0, 1]$
    `Si` $rw \leq rw_{seuil}$ `alors`
        Choisir un mouvement m au hasard
    `Sinon`
        Choisir le meilleur mouvement m autorisé
    `FinSi`
    Mettre à jour la liste Tabou en fonction de m
    Effectuer le mouvement m dans x
    Mettre à jour l'ensemble des solutions non dominées ND en fonction de x
`FinPour`
Renvoyer ND

Fixer rw_{seuil} à 0 conduit à un algorithme standard *TS* alors que fixer rw_{seuil} à 1 donne un algorithme de recherche uniquement aléatoire.

L'algorithme Tabou basé sur une mesure de la distance de Hamming *TS+HW*.

Notre troisième et dernier algorithme Tabou est basé sur une technique de diversification originale et plus rationnelle (par rapport à *TS+RW*). L'idée principale consiste à superviser l'évolution de la diversité des configurations récemment visitées. Si le niveau de diversité tombe sous un certain seuil, une phase de diversification est exécutée.

La diversité des configurations est calculée en utilisant la *distance de Hamming*. Pour appliquer cette idée, il est nécessaire de pouvoir calculer efficacement et de manière incrémentale la distance de Hamming sur l'ensemble des configurations données. La formule, dédiée ainsi qu'une technique incrémentale, sont décrites à la Section 4.3.

À chaque itération de l'algorithme *TS+HW*, la diversité des l dernières configurations (l est fixé de manière empirique) est calculée. Si la valeur de la diversité est inférieure à un certain seuil d, l'algorithme change de comportement et entame une phase de diversification. Il existe plusieurs manières de réaliser cette phase de diversification. Dans cette thèse, nous avons juste augmenté la valeur Tabou des mouvements les plus fréquents, puis redémarré la recherche depuis une configuration détériorée (par exemple, la configuration dans laquelle toutes les variables sont mises à 0). L'algorithme *TS+HW* est décrit dans l'algorithme 7.

Algorithme 7 L'algorithme Tabou utilisant la distance de Hamming *TS+HW*.

Soit x une configuration réalisable, L le nombre d'itérations, et d le seuil de diversité
$ND \leftarrow \{x\}$ *(ND est l'ensemble des solutions non dominées)*
`Pour` $i = 0$ `jusqu'à` L `faire`
    `Si` $Niveau\_de\_diversité < d$ `alors`
        Augmenter la valeur Tabou des mouvements les plus fréquemment
        effectués durant la dernière phase d'intensification
        $x \leftarrow$ configuration-zéro
    `FinSi`
    Choisir le meilleur mouvement m autorisé
    Mettre à jour la liste Tabou en fonction de m
    Effectuer le mouvement m dans x
    Mettre à jour l'ensemble des solutions non dominées ND en fonction de x
`FinPour`
Renvoyer ND

4.3 Distance de Hamming et évaluation incrémentale.

La distance de Hamming entre deux configurations $x = (x_1, x_2, ..., x_n)$ et $y = (y_1, y_2, ..., y_n)$ est définie de la manière suivante :

$$\sum_{i=1}^{n} |x_i - y_i|, \qquad x_i, y_i \in \{0, 1\}$$

Plus généralement, la somme des distances de Hamming SH d'un ensemble de p configurations avec n variables est définie ainsi :

$$SH = \sum_{j=1}^{p-1} \sum_{j'=j+1}^{p} \sum_{i=1}^{n} |y_{ij} - y_{ij'}|, \quad y_{ij}, y_{ij'} \in \{0, 1\}$$

Ainsi, la valeur SH représente la somme des distances de Hamming des p configurations prises deux à deux. Cette mesure est calculable dans un temps quadratique dépendant de p. Ce calcul n'est donc pas facilement utilisable pour de très grandes valeurs de p ou lorsqu'un nombre très important de mesures doit être fait.

Récemment, R.W Morrison et K.A De Jong [Morrison and De Jong, 2001] ont proposé une transformation de la formule classique. La nouvelle formule est maintenant calculable en temps linéaire. Elle est définie par :

$$\text{Soit} \quad w_i = \frac{\sum_{t=1}^{p} x_{it}}{p}$$
$$\text{alors} \quad SH = p \times \left(\sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - w_i)^2 \right)$$

Cette formule linéaire n'est cependant pas sous une forme incrémentale. En remplaçant w_i par son expression, et en développant la formule, nous obtenons le raisonnement développé ci-après :

$$SH = p \times \left(\sum_{i=1}^{n} \sum_{j=1}^{p} \left(x_{ij} - \frac{\sum_{t=1}^{p} x_{it}}{p} \right)^2 \right)$$

$$SH = p \times \left(\sum_{i=1}^{n} \sum_{j=1}^{p} \left(x_{ij}^2 - \frac{2}{p} \times x_{ij} \times \sum_{t=1}^{p} x_{it} + \left(\frac{\sum_{t=1}^{p} x_{it}}{p} \right)^2 \right) \right)$$

$$\text{soit } SH = p \times \sum_{i=1}^{n} \left(\sum_{j=1}^{p} x_{ij}^2 - \frac{2}{p} \times \left(\sum_{t=1}^{p} x_{it} \right) \times \left(\sum_{j=1}^{p} x_{ij} \right) + \sum_{j=1}^{p} \left(\frac{\sum_{t=1}^{p} x_{it}}{p} \right)^2 \right)$$

En effectuant un changement de variable, nous obtenons :

$$SH = p \times \sum_{i=1}^{n} \left(\sum_{j=1}^{p} x_{ij}^2 - \frac{2}{p} \times \left(\sum_{j=1}^{p} x_{ij} \right)^2 + \frac{p}{p^2} \times \left(\sum_{j=1}^{p} x_{ij} \right)^2 \right)$$

comme $x_{ij} \in \{0, 1\}$, nous avons $x_{ij}^2 = x_{ij}$

$$\text{et} \quad SH = p \times \sum_{i=1}^{n} \left(\sum_{j=1}^{p} x_{ij} - \frac{1}{p} \times \left(\sum_{j=1}^{p} x_{ij} \right)^2 \right)$$

$$\text{soit } SH = \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{p} x_{ij} \right) \times \left(p - \sum_{j=1}^{p} x_{ij} \right) \right)$$

□

Sous cette forme, SH devient incrémentale et peut donc être intégrée efficacement dans un processus incrémental.

4.4 Étude expérimentale du rôle de la diversité

Dans cette section, nous effectuons une étude expérimentale des trois algorithmes Tabou présentés précédemment, afin de mettre en valeur l'importance de la diversification dans le cadre de l'optimisation multiobjectif.

4.4.1 Les jeux de test

Les expérimentations sont effectuées sur neuf instances du MOKP publiées dans [Zitzler and Thiele, 1999]. Ces jeux d'essai ont 2, 3 et 4 objectifs combinés avec 250, 500 et 750 objets. De plus, pour chaque instance, le nombre de contraintes est égal au nombre

d'objectifs. Ces instances ont été construites grâce à un générateur aléatoire. Les poids et les profits sont décorrélés, et les capacités de chaque contrainte sont réglées à environ la moitié de la somme des poids de la contrainte considérée. Il en découle que le nombre d'objets présents attendu dans les solutions optimales est d'environ la moitié du nombre total d'objets du problème.

Dans ce chapitre, nous n'utilisons que les instances bi-objectifs. Au delà de 2 objectifs, il est difficile de représenter graphiquement les surfaces de compromis. Pour notre étude, nous nous restreignons aux instances dont les résultats sont représentables graphiquement.

4.4.2 Paramètres expérimentaux

Tous les algorithmes Tabou sont programmés en `CAML` [2]et compilés avec `OCAML` [2]. Dans le but d'obtenir les comparaisons les plus justes, nous partageons les structures de données les plus importantes entre les différents algorithmes.

Dans nos expérimentations, nous choisissons de manière empirique les paramètres suivants :

- Pour *TS+RW*, la valeur seuil pour la marche aléatoire rw_{seuil} est fixée à 0.15,
- Pour *TS+HW*, la valeur seuil pour le niveau de diversité d est fixée à 0.15.

Le choix des valeurs des paramètres est effectué en expérimentant l'algorithme sur un petit jeu d'essai (ici l'instance MOKP à 250 objets, 2 objectifs et 2 contraintes) pour différentes valeurs de ces paramètres. Nous conservons les valeurs permettant à l'algorithme d'obtenir les meilleurs résultats. Cette méthode est utilisée pour la détermination de tous les paramètres choisis dans cette thèse.

Le tableau 4.1 récapitule le réglage des principaux paramètres utilisés par *TS*, *TS+RW* et *TS+HW*. Le tableau 4.2 indique, pour chacun des trois algorithmes en compétition et pour chaque instance du problème, le temps de calcul obtenu en fonction des paramètres choisis [3]. Chaque instance de problème étant résolue dix fois par chaque algorithme, les valeurs du tableau 4.2 représentent la moyenne du temps de calcul sur les dix exécutions. Les algorithmes utilisés ici comprennent tous des opérateurs stochastiques [4]. Nous exécutons donc dix fois chaque algorithme et réalisons la moyenne, pour chaque mesure, des résultats obtenus sur les différentes exécutions. Ainsi, aucun paramètre n'est modifié entre chaque exécution. Dans la suite de cette thèse, il est sous-entendu que lorsque dix exécutions sont effectuées, aucune modification des paramètres initiaux n'est appliquée.

4.4.3 Comparaisons

Dans cette sous-section, nous présentons les résultats expérimentaux des trois algorithmes de recherche Tabou. Les résultats décrits ci-après sont calculés en réalisant la

2. `CAML` est un acronyme de "Categorical Abstract Machine Language". C'est un langage fonctionnel à la `LISP`, téléchargeable à l'adresse : `http://caml.inria.fr`

3. Les temps de calcul sont fonctions du code binaire, généré par le compilateur `OCAML`, exécuté sur un PC sous Linux (Bi-Pentium III 1 Ghz).

4. Le générateur de séquences de nombres aléatoires étant initialisé sur l'horloge de la machine, deux exécutions de nos algorithmes donnent des résultats légèrement différents.

Instance		Algorithme	
Nombre d'objectifs	Nombre d'objets	Taille de la liste Tabou (pour tous les algorithmes)	Nombre d'itérations (pour tous les algorithmes)
2	250	19	100 000
	500	21	200 000
	750	23	400 000

TAB. 4.1 – *Réglages des paramètres pour les différents algorithmes en fonction de l'instance traitée.*

Nombre d'objectifs	Nombre d'objets	*TS*	*TS+RW*	*TS+HW*
2	250	27	24.86	37.96
	500	113.32	105.58	151.88
	750	336.26	304.96	456.67

TAB. 4.2 – *Moyenne des temps de calcul obtenus par chaque algorithme (secondes).*

moyenne sur dix exécutions indépendantes.

Dans le tableau 4.3 se trouvent les résultats obtenus pour la mesure $\mathcal{H}$ (cf. Section 3.3). Nous observons que les deux algorithmes *TS+HW* et *TS+RW* donnent de meilleurs résultats (plus grandes valeurs de $\mathcal{H}$) que *TS* pour toutes les instances. En comparant *TS+HW* et *TS+RW*, nous observons que *TS+HW* obtient de meilleurs résultats que *TS+RW* sur toutes les instances, avec un plus grand écart qu'entre *TS* et *TS+RW*. Il est intéressant de noter les très bons résultats de *TS+HW* sur les deux plus grandes instances.

Nous avons observé, lors de nos expérimentations, qu'en augmentant le nombre d'itérations de nos algorithmes Tabou, nous améliorons la qualité des résultats. Cependant, le rapport (*qualité*/*nombre_itérations*) décroît de manière très importante.

La figure 4.1 récapitule les résultats de la mesure $\mathcal{C}$ (cf. Section 3.4) dans le même esprit que [Zitzler and Thiele, 1999]. Sur cette figure, chaque barre noire représente les résultats, mesurés avec $\mathcal{C}$, obtenus entre deux algorithmes sur une instance du problème. Deux valeurs sont représentées : l'écart entre les valeurs maximales et minimales obtenues sur les dix exécutions, et la moyenne de $\mathcal{C}$ calculée sur les toutes les exécutions.

Nombre d'objectifs	Nombre d'objets	*TS*	*TS+RW*	*TS+HW*
2	250	8.97e+7	9.02e+7	9.84e+7
	500	3.72e+8	3.74e+8	4.06e+8
	750	7.96e+8	8.04e+8	8.88e+8

TAB. 4.3 – *Moyenne de l'hypervolume* $\mathcal{H}$.

Il est clair qu'une fois encore, les résultats obtenus par *TS* sont inférieurs à ceux de *TS+RW* et *TS+HW* sur toutes les instances. En comparant *TS+RW* et *TS+HW*, nous constatons que les résultats sont en faveur de *TS+HW* pour chacune des instances. En effet, les valeurs obtenues par $\mathcal{C}(\textit{TS+RW}, \textit{TS+HW})$ sont inférieures à celles obtenues par $\mathcal{C}(\textit{TS+HW}, \textit{TS+RW})$. Il est intéressant de noter que sur les instances les plus difficiles (500 et 750 objets), le facteur séparant les différents résultats est supérieur à 10.

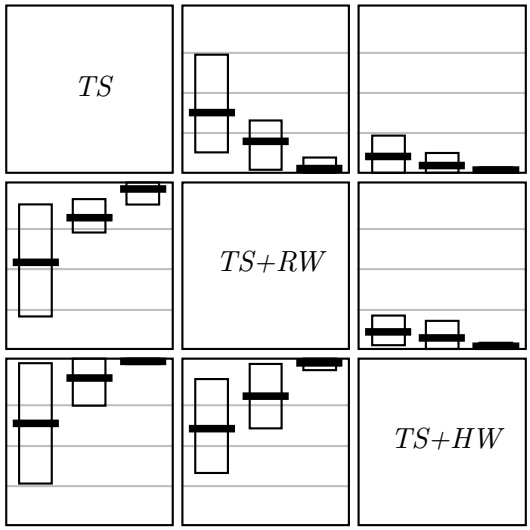
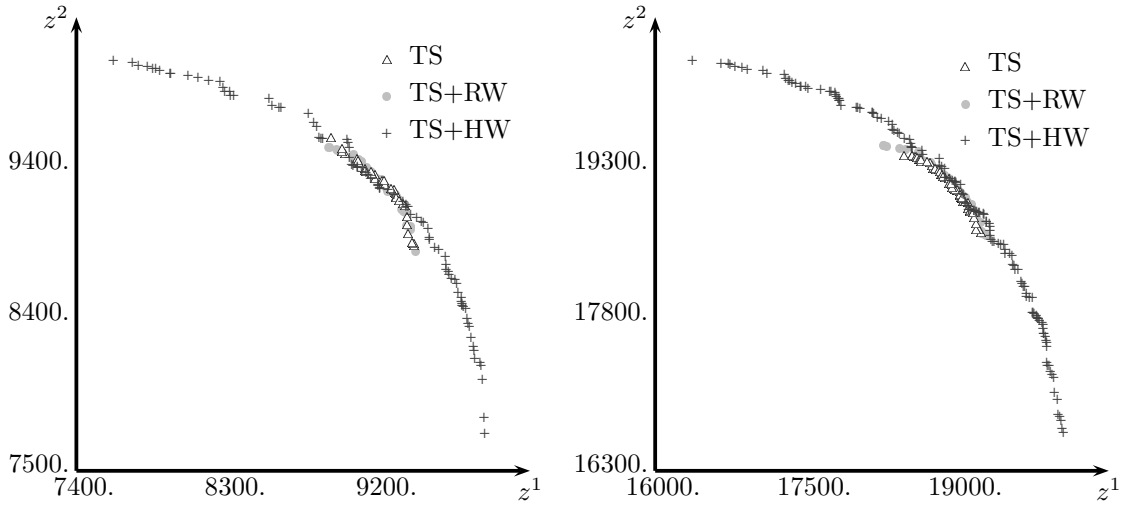


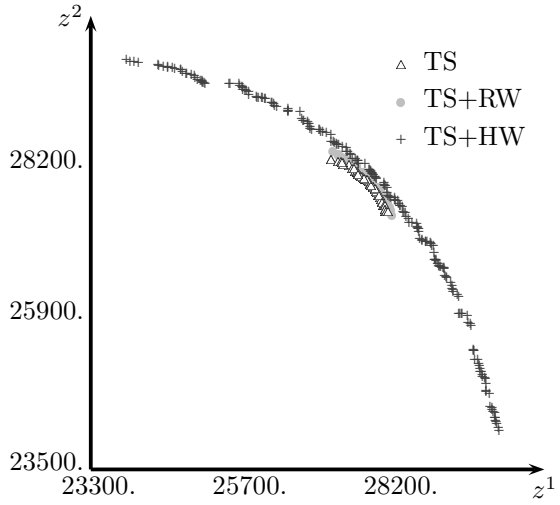
FIG. 4.1 – *Résultats obtenus par* TS, TS+RW *et* TS+HW, *évalués par la mesure* $\mathcal{C}$. Chaque boîte correspond à la comparaison de l'algorithme A en ligne avec l'algorithme B en colonne. Sachant que l'échelle part de 0 en bas de la boîte pour aller à 1 en haut de celle-ci, chaque rectangle représente la répartition des résultats de $\mathcal{C}(A, B)$ des différentes exécutions. Chaque barre horizontale noire correspond à la moyenne des différentes mesures de $\mathcal{C}(A, B)$. Chaque boîte contient trois rectangles (liés à trois barres noires) correspondant respectivement, de gauche à droite, aux instances contenant 250, 500 et 750 objets.

La figure 4.2 montre la surface de compromis procurée par chaque algorithme. La hiérarchie mise en valeur par les précédents résultats est respectée sur ce graphique. Ainsi, la surface décrite par *TS* est la plus étriquée sur toutes les instances. La surface décrite par *TS+RW* occupe la *seconde place*, la surface décrite par *TS+HW* étant clairement meilleure que celles de *TS* et *TS+RW*. Certaines portions des trois surfaces se superposant, leur étude est plus difficile. Dans ces zones, les trois algorithmes obtiennent des résultats comparables. Finalement, il est intéressant de noter que plus les instances augmentent en taille et en difficulté, et plus le comportement de *TS+HW* est performant en comparaison de *TS* et *TS+RW*.



Instance avec 250 objets

Instance avec 500 objets



Instance avec 750 objets

FIG. 4.2 – *Surfaces de compromis procurées par les trois algorithmes Tabou sur trois instances du problème du sac à dos.*

4.4.4 Discussion

Les résultats obtenus par l'algorithme *TS+HW* sont meilleurs que ceux obtenus par *TS+RW* et *TS*. Les résultats de ces deux derniers s'améliorent lorsque nous augmentons le nombre d'itérations, mais la surface de compromis obtenue est toujours de qualité inférieure à celle de *TS+HW*. L'algorithme *TS+HW* explore mieux l'espace de recherche.

Il apparaît que le mécanisme de diversification, composé uniquement par la liste Tabou ou amélioré par une marche aléatoire, ne suffit par pour réaliser une recherche efficace. En complément, le mécanisme de diversification basé sur la mesure de la distance de Hamming, utilisé par *TS+HW*, permet l'obtention de meilleurs résultats. Ce mécanisme permet à l'algorithme de diversifier sa recherche quand cela est nécessaire, et aide de plus à ajuster les valeurs Tabou de manière dynamique et auto-adaptative.

Au chapitre 5, nous verrons un exemple d'algorithme hybride combinant une méthode évolutionnaire performante avec l'algorithme *TS* vu précédemment. Dans cet algorithme appelé GTS^{MOKP}, la diversification de la recherche est assurée par la population et le mécanisme de sélection. Assurer la diversification grâce aux actions combinées de la population et de la sélection est très courant chez les algorithmes évolutionnaires. Cette aptitude naturelle à assurer une diversification de la recherche est sans doute une des clefs de leur succès.

4.5 Comparaisons avec deux algorithmes évolutionnaires

Dans cette section, nous comparons notre meilleur algorithme Tabou, c.à.d *TS+HW*, avec *SPEA* [Zitzler and Thiele, 1999] et *MOGLS* [Jaszkiewicz, 2002], deux algorithmes de référence pour le MOKP.

4.5.1 Paramètres expérimentaux

Comme *TS+HW*, *SPEA* et *MOGLS* sont programmés [5] dans le langage `CAML` et compilés avec `OCAML`. Là aussi, les structures de données les plus importantes sont partagées entre les trois algorithmes, dans le but d'obtenir une base aussi solide que possible pour nos comparaisons.

Dans nos expérimentations, nous utilisons les paramètres suivants :

- Pour *TS+HW*, la valeur seuil pour le niveau de diversité d est fixée à 0.15,
- Pour *SPEA*, Nous fixons le taux de mutation à 0.2, et le niveau de "clustering" à 15000 (cf. [Zitzler and Thiele, 1999] et [Jaszkiewicz, 2000b]),
- La taille de la population pour *SPEA* et *MOGLS* est fixée en fonction du nombre d'objets et du nombre d'objectifs de l'instance. En revanche, pour une même instance, les algorithmes utilisent le même paramétrage (cf tableau 4.4).

5. Les implémentations de *MOGLS* et *SPEA* utilisées ici sont calquées sur celles de la MOMHLib [Jaszkiewicz, 2000a].

Instance		Algorithme					
Nombre d'objectifs	Nombre d'objets	Taille de la population initiale		Nombre de générations		Nombre d'itérations	Taille de la liste Tabou
		SPEA	*MOGLS*	*SPEA*	*MOGLS*	*TS+HW*	*TS+HW*
2	250	150	150	50	50	50000	19
	500	200	200	50	50	55000	21
	750	250	250	50	50	60000	23

TAB. 4.4 – *Réglages des paramètres pour les différents algorithmes en fonction de l'instance traitée.*

Nombre d'objectifs	Nombre d'objets	*SPEA*	*MOGLS*	*TS+HW*
2	250	7.48	23.41	18.50
	500	25.66	48.64	43.95
	750	56.16	82.33	71.87

TAB. 4.5 – *Moyenne des temps de calcul obtenus par chaque algorithme (secondes).*

Le tableau 4.4 récapitule le réglage des principaux paramètres utilisés par *TS+HW*, *SPEA* et *MOGLS*.

Le tableau 4.5 indique, pour chacun des trois algorithmes en compétition et pour chaque instance du problème, le temps de calcul obtenu en fonction des paramètres choisis. Au regard de ce tableau, il apparaît que *SPEA* est l'algorithme en compétition le plus rapide quelle que soit l'instance, suivi de *TS+HW* puis de *MOGLS*. Toutefois, notons que les différents temps de calcul ne s'élèvent pas à plus d'une centaine de secondes, et que les écarts entre les différents algorithmes ne sont pas significatifs sur un si court laps de temps.

4.5.2 Comparaisons entre *TS+HW*, *SPEA* et *MOGLS*

Le tableau 4.6 montre la moyenne des résultats obtenus pour la mesure $\mathcal{H}$. Il est clair que *TS+HW* et *MOGLS* obtiennent de meilleurs résultats (plus grandes valeurs de $\mathcal{H}$) que *SPEA* pour toutes les instances. En comparant *TS+HW* et *MOGLS*, nous observons que *MOGLS* obtient des résultats légèrement supérieurs sur toutes les instances, mais l'écart entre ces deux algorithmes est inférieur à celui séparant *TS+HW* de *SPEA*.

Comme pour la section 4.4.3, la figure 4.3 récapitule les résultats de la mesure $\mathcal{C}$. Il est clair que les résultats obtenus par *MOGLS* sont supérieurs à ceux obtenus par *SPEA* et *TS+HW* sur toutes les instances. En comparant *SPEA* et *TS+HW*, nous constatons que *TS+HW* est aussi bon que *SPEA* sur la première instance. Pour les deux dernières, les résultats sont clairement en faveur de *TS+HW*, en effet, les valeurs obtenues par $\mathcal{C}$(*SPEA*, *TS+HW*) sont inférieures aux valeurs obtenues par $\mathcal{C}$(*TS+HW*, *SPEA*). Il est important de noter que, sur les instances les plus difficiles (500 et 750 objets), les résultats de *TS+HW* sont nettement supérieurs à ceux de *SPEA*.

Nombre d'objectifs	Nombre d'objets	*SPEA*	*MOGLS*	*TS+HW*
2	250	9.40e+7	9.86e+7	9.84e+7
	500	3.82e+8	4.07e+8	4.05e+8
	750	8.06e+8	8.92e+8	8.77e+8

TAB. 4.6 – *Moyenne de l'hypervolume $\mathcal{H}$.*

Les surfaces de compromis procurées par chaque algorithme sont représentées à la figure 4.4. Il apparaît clairement que *MOGLS* obtient la meilleure surface de compromis et confirme donc les résultats observés précédemment. Cependant, sur l'instance la plus difficile (750 objets), des points découverts par *TS+HW* dominent ceux calculés par *MOGLS* dans certaines zones de l'espace des objectifs.

TS+HW semble améliorer ses performances sur les instances les plus difficiles, comparativement à *MOGLS* et *SPEA*. Sur ces instances, *TS+HW* est capable de découvrir des solutions qui sont non dominées par celles des deux autres algorithmes. Il est aussi capable de trouver des solutions qui dominent certaines calculées par *MOGLS* et *SPEA*. En d'autres termes, *TS+HW*, *MOGLS* et *SPEA* découvrent des zones différentes de l'espace des objectifs, car ils explorent de manière différente l'espace de recherche. Cette observation nous donne une justification des approches combinant méthodes génétiques et méthodes de recherche locale en un seul algorithme [Jaszkiewicz, 2002].

4.6 Conclusion

Dans ce chapitre, nous avons présenté une étude empirique de la recherche Tabou pour le problème du sac à dos multidimensionnel multiobjectif en 0-1 (MOKP). Nous avons étudié trois algorithmes Tabou, *TS*, *TS+RW* et *TS+HW* incorporant chacun des techniques de diversification différentes. Les résultats expérimentaux ont mis en évidence les excellents résultats de l'algorithme *TS+HW*, notamment liés à son mécanisme de diversification basé sur la distance de Hamming. L'essentiel de ces travaux est publié dans [Barichard and Hao, 2002].

Les expérimentations ont aussi mis en valeur les bons résultats de *TS+HW*, en comparaison de deux algorithmes évolutionnaires réputés : *SPEA* et *MOGLS*. Même si l'algorithme Tabou n'opère qu'avec une seule solution durant la recherche, les résultats obtenus sont meilleurs que ceux de *SPEA*, et rivalisent dans certains cas avec ceux de *MOGLS*. Ceci est d'autant plus remarquable que *TS+HW* est un algorithme très simple comparé aux algorithmes utilisant une population. En effet, *SPEA* comme *MOGLS*, sont basés sur des modèles évolutionnaires complexes et intègrent déjà des méthodes d'amélioration locale des solutions.

L'algorithme *TS+HW* peut s'améliorer de plusieurs façons. Premièrement, en conservant la même technique de surveillance de la diversité, les actions menées durant la phase de diversification peuvent être perfectionnées. Deuxièmement, les mécanismes de traitement

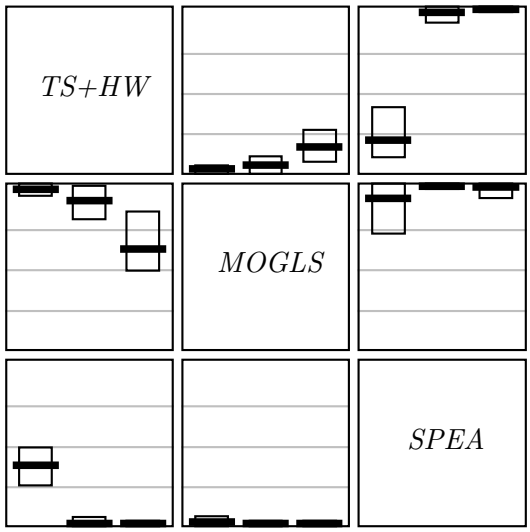
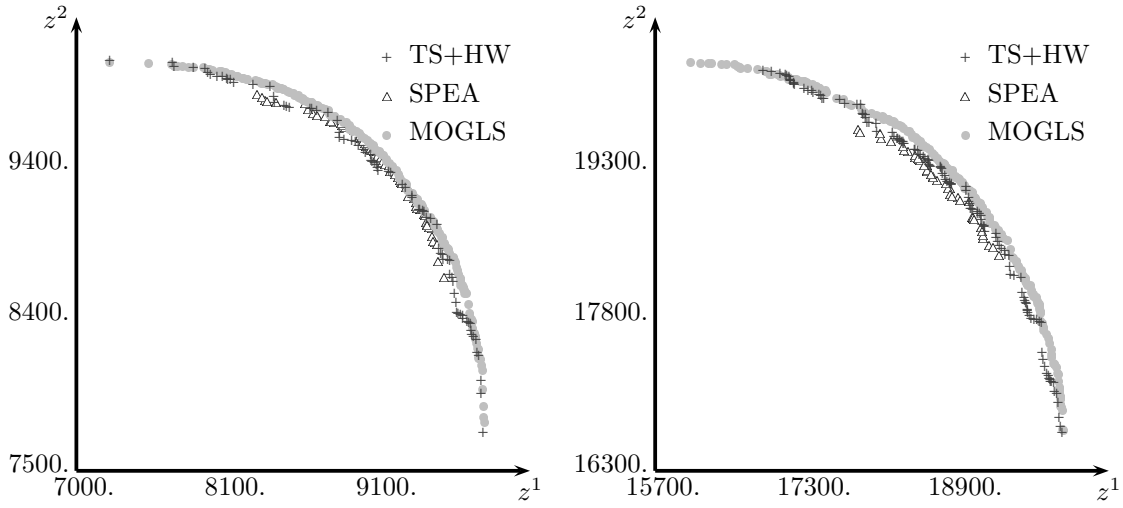


FIG. 4.3 – *Résultats obtenus par* `TS+HW`*,* `SPEA` *et* `MOGLS`*, évalués par la mesure* $\mathcal{C}$*.* Chaque boîte correspond à la comparaison de l'algorithme A en ligne avec l'algorithme B en colonne. Sachant que l'échelle part de 0 en bas de la boîte pour aller à 1 en haut de celle-ci, chaque rectangle représente la répartition des résultats de $\mathcal{C}(A, B)$ des différentes exécutions. Chaque barre horizontale noire correspond à la moyenne des différentes mesures de $\mathcal{C}(A, B)$. Chaque boîte contient trois rectangles (liés à trois barres noires) correspondant respectivement, de gauche à droite, aux instances contenant 250, 500 et 750 objets.

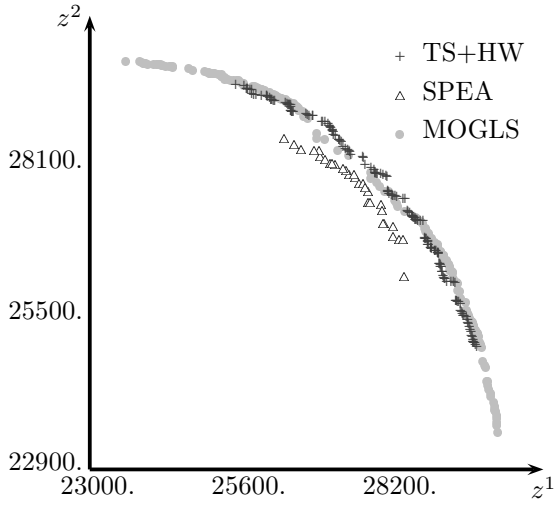
des contraintes développées pour le problème classique du sac à dos multidimensionnel en 0-1 [Vasquez and Hao, 2001] pourraient être bénéfiques dans le contexte du MOKP.

Cette étude met en évidence l'importance d'une gestion efficace de la diversité durant la recherche dans le cadre des problèmes d'optimisation multiobjectifs, et ceci reste vrai aussi bien pour les algorithmes évolutionnaires que pour les algorithmes de recherche locale. Cette étude suggère aussi que l'incorporation d'une recherche Tabou dans un cadre évolutionnaire pourrait amener à la réalisation d'un algorithme hybride très performant pour l'optimisation multiobjectif. Au chapitre 5, nous approfondissons cette voie et développons un algorithme hybride combinant une approche évolutionnaire et notre algorithme Tabou `TS`. Dans ce schéma, la diversification est assurée par les actions combinées de la population et de la sélection.



Instance avec 250 objets

Instance avec 500 objets



Instance avec 750 objets

FIG. 4.4 – *Surfaces de compromis procurées par les trois algorithmes en compétition sur trois instances du problème du sac à dos.*

Chapitre 5

Un algorithme hybride pour le problème de sac à dos multiobjectif

Dans ce chapitre, nous nous intéressons à l'approche hybride et présentons GTS^{MOKP}, un algorithme génétique combiné à une recherche Tabou pour le MOKP. Inspiré par un algorithme hybride pour le problème de coloriage de graphe [Galinier and Hao, 1999], et par l'algorithme *MOGLS* [Jaszkiewicz, 2000b], GTS^{MOKP} se distingue de *MOGLS* essentiellement par l'intégration d'une méthode de recherche de voisinage très performante (l'algorithme Tabou *TS*) au schéma génétique. Au travers d'expérimentations menées avec GTS^{MOKP}, nous montrons que l'incorporation d'une méthode d'intensification efficace (ici l'algorithme *TS*) influe sur la qualité des résultats. L'essentiel des travaux effectués dans ce chapitre est publié dans [Barichard and Hao, 2003a].

Sommaire

5.1 Rappel du contexte

Dans ce chapitre, nous utilisons le même cadre de travail qu'au chapitre 4. Nous rappelons la formulation du sac à dos multidimensionnel multiobjectif présenté à la Section 4.1. Soit un ensemble d'objets, à chacun étant associé un vecteur de profits et de poids. Le problème de sac à dos multidimensionnel multiobjectif (MOKP) consiste à sélectionner un sous-ensemble d'objets maximisant une fonction multiobjectif tout en satisfaisant un ensemble de contraintes. Plus formellement, le MOKP peut être défini comme suit :

$$\text{MOKP01} \begin{cases} \max & z^j(x) = \sum_{i=1}^{n} c_i^j x_i & j = 1, ..., m \\ \text{s.t.} & \sum_{i=1}^{n} w_i^l x_i \leq b_l & l = 1, ..., q \\ & x_i \in \{0, 1\} & i = 1, ..., n \end{cases}$$

où n est le nombre d'objets, x_i une variable de décision, m le nombre d'objectifs, z^j la $j^{ème}$ composante de la fonction multiobjectif z, et q le nombre de contraintes du problème.

Au chapitre 4, nous avons vu que la diversification de la recherche joue un rôle important dans la résolution des problèmes d'optimisation multiobjectifs. Dans ce chapitre, nous montrons que l'intensification, bien qu'étant une action opposée à la diversification, influe grandement sur la qualité des résultats. Les algorithmes évolutionnaires effectuent naturellement des phases de diversification, pour peu que le processus de sélection soit choisi de manière adéquate. La recherche Tabou est une méthode bien connue pour son aptitude à intensifier la recherche. Pour observer le rôle de l'intensification, nous avons choisi de partir d'un algorithme évolutionnaire performant et de lui greffer une méthode d'intensification puissante. Dans ce but, nous développons un nouvel algorithme pour le MOKP. Cet algorithme, appelé GTS^{MOKP}, combine une approche évolutionnaire et une méthode Tabou. Dans ce schéma, inspiré par un algorithme hybride pour le problème de coloriage de graphe [Galinier and Hao, 1999], la diversification est assurée par les actions combinées de la population et de la sélection, l'opérateur de recherche Tabou prenant en charge les phases d'intensification. Parmi toutes les approches proposées pour résoudre le MOKP, les meilleurs résultats observés, quant aux algorithmes de la littérature sur les instances que nous allons utiliser, sont détenus par l'algorithme ***MOGLS*** présenté dans [Jaszkiewicz, 2000b].

Pour évaluer les performances de GTS^{MOKP}, l'algorithme est testé sur un jeu de 9 instances publiées de MOKP, puis comparé avec ***NSGA*** et ***MOGLS***, deux algorithmes réputés. Lors des expérimentations, GTS^{MOKP} a réussi à améliorer tous les meilleurs résultats détenus jusqu'ici par ***MOGLS***.

Dans la prochaine section, nous présentons les principes généraux de notre algorithme hybride GTS^{MOKP}. Les résultats expérimentaux, ainsi que les comparaisons de GTS^{MOKP} avec les deux autres algorithmes testés, sont le sujet de la section 5.3. Dans la dernière section, nous présentons quelques conclusions.

5.2 Recherche Tabou et algorithme génétique pour le MOKP (GTS^{MOKP})

L'hybridation entre un algorithme génétique et une recherche locale est maintenant reconnue comme une approche compétitive pour traiter des problèmes combinatoires difficiles. Pour le MOKP, nous pouvons citer l'algorithme *MOGLS* utilisant une méthode de descente pure comme opérateur de recherche locale [Jaszkiewicz, 2000b].

Un principe utilisé pour hybrider un algorithme génétique et une méthode de recherche locale consiste à remplacer l'opérateur de mutation de l'algorithme génétique par un opérateur de recherche locale. À chaque génération, un nombre de croisements (avec un opérateur de croisement spécifique au problème, ou aléatoire) est effectué. Pour chaque nouvelle configuration (individu) s ainsi générée, un opérateur de recherche locale $LS(s)$ est appliqué à s pour améliorer sa qualité. Finalement, un mécanisme de sélection décide si le nouvel individu amélioré doit être introduit dans la population.

Dans cette section, nous présentons notre algorithme hybride GTS^{MOKP} pour le MOKP.

5.2.1 Algorithme général

GTS^{MOKP} suit le schéma classique utilisé pour l'hybridation entre algorithme génétique et recherche locale. Nous donnons le pseudo-code de GTS^{MOKP} à l'algorithme 8. Plus de détails sont fournis dans la section 5.2.2.

Algorithme 8 GTS^{MOKP} : *"genetic tabu search algorithm"* pour le MOKP.

$P \leftarrow$ Init_Population ($|P|$)
`Tant que` *critère_d'arrêt_non_rencontré* `faire`
  $\lambda \leftarrow$ Vecteur_Aléatoire (utilisé pour l'évaluation, voir prochaine section)
  Evaluer chaque individu s dans P selon $f(s, r, \lambda)$ (voir prochaine section)
  $TP \leftarrow$ les N "meilleurs" individus de P
  $(p_1, p_2) \leftarrow$ Sélection_Parents(TP)
  $s \leftarrow$ Croisement(p_1, p_2)
  $s \leftarrow$ Recherche_Tabou($s, Number\_Of\_Iterations$)
  $P \leftarrow$ Ajoute_à_Population(s, P)
`FinTantQue`
Renvoyer P

Remarquons toutefois que, bien que non clairement explicitées dans l'algorithme, les solutions non-dominées sont enregistrées dans une structure de données appropriée. C'est cet ensemble de solutions qui est renvoyé en sortie de l'algorithme. Nous présentons dans la prochaine section les différents composants de GTS^{MOKP}.

5.2.2 Espace de recherche et valeurs d'évaluation

Une *configuration* (ou un *individu*) s est un vecteur binaire de n composantes satisfaisant toutes les q contraintes du problème. *L'espace de recherche* S est alors constitué par l'ensemble de tous les vecteurs binaires, qui est clairement un sous-ensemble de $\{0,1\}^n$.

Pour évaluer une configuration $s \in S$, nous utilisons, comme dans [Jaszkiewicz, 2000a], une fonction d'agrégation linéaire pondérée, définie comme suit :

$$f(s, r, \lambda) = \sum_{i=1}^{m} \lambda_i * (r_i - z_i(s)) \tag{5.1}$$

où

- s est la configuration à évaluer,
- $z_i(s)$ $(i = 1 \cdots m)$ est la valeur de s pour la $i^{ème}$ composante de la fonction objectif du problème,
- λ est un vecteur de poids composé de m composantes comprises dans l'interval $[0 \cdots 1]$, dont la $i^{ème}$ composante λ_i correspond au poids associé au $i^{ème}$ objectif,
- r est un vecteur de référence à m-composantes. Ce point de référence est dans notre cas calculé sur chaque objectif, en fonction des valeurs maximales des points de l'ensemble courant des solutions non dominées : $r_i = \max_{x \in Set}(x_i)$ $i \in [1 \cdots m]$.

Ainsi, pour chaque configuration s et pour un vecteur de référence r donné, une valeur d'adaptation ("fitness") est attribuée à s selon l'évaluation et le poids associé à chaque objectif de s. Cette fonction d'évaluation exprimée en (5.1) définit un ordre total pour les configurations de l'espace de recherche S. Cet ordre est utilisé comme base pour les opérateurs génétiques et ceux de recherche locale.

Notre fonction d'évaluation étant basée sur une fonction d'agrégation linéaire, certaines solutions Pareto optimales peuvent ne jamais être atteintes. Notons toutefois que grâce au hasard de la recherche heuristique, certaines de ces solutions peuvent quand même être déterminées par notre algorithme.

D'autres fonctions d'évaluation sont aussi possibles, comme celles basées sur le "ranking" des solutions, l'ordre lexicographique des objectifs, ... (voir [Ehrgott and Gandibleux, 2000]).

5.2.3 La recherche génétique

Le codage des individus dans GTS^{MOKP} est réalisé à partir de chaînes 0/1 de longueur n, où chaque valeur correspond à la présence ou l'absence d'un objet du sac à dos. Ainsi, changer la valeur d'une variable de 0 à 1 correspond à ajouter un objet au sac à dos, et changer la valeur d'une variable de 1 à 0 correspond à retirer un objet au sac à dos.

La première étape dans une recherche génétique est de constituer une population initiale servant de départ à l'algorithme. Pour construire sa population initiale, GTS^{MOKP} construit aléatoirement autant d'individus qu'il y a de places dans la population. Pour

construire chaque individu aléatoirement, une procédure ajoute à l'individu un sous-ensemble d'objets pris au hasard. Si la somme des poids de tous les objets excède la capacité d'une des contraintes de sac à dos, alors les objets influant le moins sur le profit sont retirés de l'individu, et ceci tant qu'il reste des contraintes violées. Puis la recherche génétique entre dans sa procédure principale, qui va être itérée tant que le critère d'arrêt spécifié au départ n'est pas rencontré.

Lors d'une itération, la première action effectuée par l'algorithme est de générer un vecteur λ. Ce vecteur de dimension m correspond à une direction dans l'espace des objectifs. Cette direction sera utilisée pour évaluer les différents individus de la population P et calculer leur valeur d'adaptation. La fonction d'évaluation 5.1 utilise en effet λ pour calculer la valeur d'adaptation de l'individu. Cette direction est essentielle lors de l'optimisation de problèmes multiobjectifs, car c'est grâce à elle que tout front Pareto est découvert par l'algorithme. S'il n'était pas possible de faire varier la direction, notre fonction d'évaluation aurait beaucoup de difficultés pour favoriser certains individus faisant pourtant partie du front Pareto. Une fois que tous les individus ont été évalués, la population P est triée en ordre décroissant, en fonction de la valeur d'adaptation des individus.

L'opérateur de croisement utilisé dans la partie génétique de GTS^{MOKP} opère sur un sous-ensemble de la population P. Pour construire ce sous-ensemble, les N premiers (meilleurs) individus [1] sont dupliqués dans une population temporaire. Puis, deux individus (de cette population temporaire) sont sélectionnés de manière aléatoire pour être recombinés grâce à un opérateur de croisement. Ainsi, les croisements ne s'effectueront que sur des individus faisant partie de cette population temporaire.

Pour effectuer le croisement, nous utilisons un opérateur classique de croisement aléatoire mono-point. Comme deux individus peuvent être issus de ce type de croisement, nous ne conservons que l'individu composé de la partie gauche du premier parent et de la partie droite du deuxième parent. L'individu résultant d'un croisement peut comporter beaucoup plus d'objets que chaque parent pris séparément et peut donc violer des contraintes. Nous restaurons donc la faisabilité de cet individu de la même manière que lors de la construction de la population initiale.

Après avoir restauré la faisabilité du nouvel individu obtenu, celui-ci est amélioré par une recherche Tabou (voir prochaine section). L'application d'une recherche locale a pour but d'améliorer l'individu le remplaçant par un autre individu s_*, encore meilleur, issu de son voisinage.

Enfin, il faut sélectionner ou pas l'individu s_* et décider si celui-ci fera partie de la prochaine génération. Pour savoir si s_* doit être conservé, nous le comparons avec le plus mauvais individu de la population temporaire. Si s_* est meilleur que le plus mauvais individu de la population temporaire, alors s_* est ajouté à la population courante et remplace l'individu le plus anciennement stocké dans celle-ci. Dans les autres cas, s_* est rejeté.

Dans GTS^{MOKP}, un seul croisement est effectué par génération. Cette particularité se retrouve dans d'autres algorithmes génétiques, comme *MOGLS*. Elle permet une évolution régulière et mieux contrôlée de la population, même si beaucoup d'opérations

1. $N \leq |P|$ est à fixer de manière empirique. Dans ce document, $N = 20$ pour $|P|$ variant de 150 à 350.

(tris, évaluations, ...) sont nécessaires pour n'effectuer en fait qu'une seule recombinaison.

5.2.4 La recherche Tabou (TS)

L'opérateur TS *Recherche Tabou(s,L)* a pour but d'améliorer une configuration réalisable s produite par l'opérateur de croisement, pour un nombre maximum d'itérations L. Cette recherche est effectuée avant d'insérer l'individu s amélioré dans la population.

Les fonctions de *voisinage* et de *mouvement* sont les mêmes que celles présentées à la Section 4.2.1. La *liste Tabou* est implémentée de manière classique, et sa taille est fixée à 2 au début de l'algorithme.

L'opérateur de recherche Tabou utilisé pour notre hybridation est en définitive l'algorithme *TS* présenté à la Section 4.2.2 du chapitre 4, dont le fonctionnement est rappelé à l'algorithme 9.

Algorithme 9 L'algorithme de recherche Tabou.

Soit s_0 une configuration réalisable, et L le nombre d'itérations
$s \leftarrow s_0$
`Pour` $i = 0$ `jusqu'à` L `faire`
    Choisir le meilleur mouvement autorisé
    Mettre à jour la liste Tabou
    Effectuer le mouvement sélectionné dans s
    Mettre à jour l'ensemble des solutions non-dominées avec s
`FinPour`
Renvoyer s

5.3 Résultats expérimentaux

Dans cette section, nous comparons notre algorithme GTS^{MOKP} avec *NSGA* et *MOGLS*, deux algorithmes réputés pour le MOKP. Les expérimentations sont effectuées sur la totalité des neuf instances présentées à la sous-section 4.4.1.

5.3.1 Paramètres expérimentaux

L'algorithme GTS^{MOKP} est programmé en `CAML` et compilé avec `OCAML`. Pour obtenir des comparaisons équitables avec les autres algorithmes de l'état de l'art, nous avons aussi implémenté, toujours en `CAML`, *NSGA* et *MOGLS*. Dans notre implémentation, les principales structures de données sont partagées par les trois algorithmes. Nous obtenons ainsi une base solide pour des comparaisons équitables entre ces algorithmes.

Avant de comparer GTS^{MOKP}, *NSGA* et *MOGLS*, nous avons d'abord comparé notre implémentation de *NSGA* et *MOGLS* avec celles de MOMHLib [Jaszkiewicz, 2000a]. Les résultats obtenus nous permettent d'affirmer que les deux implémentations ont des performances comparables.

Instances		Méthodes			
Nombre d'objectifs	Nombre d'objets	Taille de la population (tous les algorithmes)	Nombre de générations		
			NSGA	*MOGLS*	GTS^{MOKP}
2	250	150	500	110	50
	500	200	500	140	50
	750	250	500	150	50
3	250	200	1100	160	50
	500	250	1100	170	50
	750	300	1100	170	50
4	250	250	3000	300	50
	500	300	3000	320	50
	750	350	3000	320	50

TAB. 5.1 – *Valeurs des paramètres pour les différents algorithmes et instances.*

Dans nos expérimentations, nous utilisons les paramètres suivants :

- pour *NSGA*, nous initialisons le taux de mutation à 0.2, et la distance de voisinage à 0.4, en accord avec [Zitzler and Thiele, 1999] et [Jaszkiewicz, 2000b],
- pour GTS^{MOKP}, nous initialisons le nombre de générations à 50, le nombre d'itérations Tabou à $L = 12$ et la taille de la liste Tabou à 2 pour chaque exécution de l'algorithme Tabou,
- pour *NSGA* et *MOGLS*, nous autorisons toujours un nombre largement supérieur de générations pour obtenir des temps de calcul supérieurs ou égaux à ceux obtenus pour GTS^{MOKP} (voir tableau 5.1),
- L'initialisation de la taille de la population dépend du nombre d'objets ainsi que du nombre d'objectifs de l'instance traitée. Sur une même instance, la taille de la population est la même pour les trois algorithmes (voir tableau 5.1).

Notons que, pour *MOGLS* et GTS^{MOKP}, le critère d'arrêt est calculé comme suit :

$$\text{critère d'arrêt} = \text{Nombre de générations} * \text{taille de la population.}$$

La notion de génération n'est présente que pour garder la présentation classique des paramètres des algorithmes évolutionnaires.

Le tableau 5.1 récapitule les valeurs des principaux paramètres utilisés par *NSGA*, *MOGLS* et GTS^{MOKP}.

Le tableau 5.2 donne, pour chacun des trois algorithmes comparés et pour chaque instance du problème, le temps de calcul obtenu pour les valeurs des paramètres [2]. Notons que *NSGA* demande toujours plus de temps de calcul que *MOGLS*, qui lui même demande plus de temps de calcul que GTS^{MOKP}.

2. Le temps de calcul est basé sur le code généré par le compilateur `OCAML` sur un PC Bi-Pentium III 1 Ghz.

Nombre d'objectifs	Nombre d'objets	*NSGA*	*MOGLS*	GTS^{MOKP}
2	250	60	54	50
	500	205	150	135
	750	330	200	185
3	250	265	250	210
	500	900	490	465
	750	1800	740	660
4	250	1417	1264	1188
	500	3500	2300	2200
	750	7000	3100	2500

TAB. 5.2 – *Moyenne des temps de calcul pour chaque algorithme (secondes).*

Nombre d'objectifs	Nombre d'objets	*NSGA*	*MOGLS*	GTS^{MOKP}
2	250	9.52e+7	9.86e+7	9.87e+7
	500	3.91e+8	4.08e+8	4.08e+8
	750	8.36e+8	8.93e+8	8.93e+8
3	250	8.45e+11	9.33e+11	9.35e+11
	500	6.74e+12	7.72e+12	7.73e+12
	750	2.33e+13	2.71e+13	2.72e+13
4	250	6.97e+15	8.11e+15	8.12e+15
	500	1.09e+17	1.35e+17	1.36e+17
	750	5.52e+17	7.19e+17	7.20e+17

TAB. 5.3 – *Moyenne de l'hypervolume* $\mathcal{H}$.

5.3.2 Résultats comparatifs

Dans cette section, nous présentons les résultats expérimentaux obtenus par GTS^{MOKP} ainsi que ceux de *NSGA* et *MOGLS*, deux algorithmes de référence. Les algorithmes utilisés ici sont tous stochastiques. Nous exécutons donc dix fois chaque algorithme et réalisons la moyenne, pour chaque mesure, des résultats obtenus sur les différentes exécutions.

Le tableau 5.3 montre les résultats obtenus pour la mesure $\mathcal{H}$ (cf. Section 3.3). Sur ce tableau, nous observons premièrement que GTS^{MOKP} et *MOGLS* donnent tous les deux de meilleurs résultats (des valeurs de $\mathcal{H}$ plus élevées) que *NSGA*, sur toutes les instances. En comparant GTS^{MOKP} et *MOGLS*, nous observons que GTS^{MOKP} dépasse *MOGLS* pour 7 des 9 instances et donne des résultats identiques sur les 2 instances bi-objectifs restantes (500 et 750 objets). Nous remarquons aussi que GTS^{MOKP} obtient de très bons résultats sur les instances les plus difficiles.

La figure 5.1 récapitule les résultats de la mesure $\mathcal{C}$ (cf. Section 3.4) suivant la présenta-

tion adoptée dans [Zitzler and Thiele, 1999]. Sur cette figure, chaque petite barre noire représente les résultats de la mesure $\mathcal{C}$ entre deux méthodes, pour chaque instance de problème. Du fait d'une très faible dispersion des résultats, la moyenne est pertinente comme valeur retenue pour la mesure $\mathcal{C}$.

Il est observé une fois de plus, que les résultats obtenus par *NSGA* sont inférieurs à ceux de *MOGLS* et de GTS^{MOKP} pour toutes les instances. En comparant *MOGLS* et GTS^{MOKP}, nous observons que les résultats penchent en faveur de GTS^{MOKP} sur toutes les instances. En effet, les valeurs obtenues $\mathcal{C}(MOGLS, GTS^{MOKP})$ sont inférieures à celles de $\mathcal{C}(GTS^{MOKP}, MOGLS)$. Notons que, sur plusieurs instances (500 objets 3 objectifs, et 500 objets 4 objectifs) le rapport entre les résultats est supérieur à 10, ce qui signifie que la quasi-totalité des solutions trouvées par GTS^{MOKP} sont de qualité égale ou supérieure à celles trouvées par *MOGLS*.

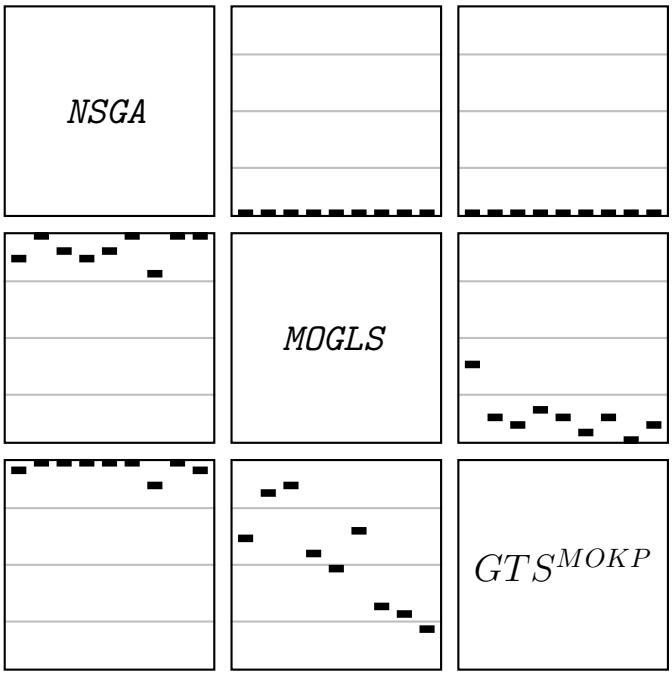


FIG. 5.1 – *Résultats des comparaisons obtenues avec la mesure* $\mathcal{C}$. Chaque boîte correspond à la comparaison de l'algorithme A en ligne avec l'algorithme B en colonne. Sachant que l'échelle part de 0 en bas de la boîte pour aller à 1 en haut de celle-ci, chaque rectangle représente la répartition des résultats de $\mathcal{C}(A, B)$ des différentes exécutions. Chaque barre horizontale noire correspond à la moyenne des différentes mesures de $\mathcal{C}(A, B)$. Chaque boîte contient trois rectangles (liés à trois barres noires) correspondant respectivement, de gauche à droite, aux instances contenant 250, 500 et 750 objets.

En résumé, même si GTS^{MOKP} est exécuté dans des conditions défavorables (moins

de générations et moins de temps de calcul autorisé), il donne des résultats au moins égaux ou supérieurs à ceux obtenus par *NSGA* et *MOGLS* sur les 9 instances testées avec les deux mesures utilisées. Pour nous convaincre de cette supériorité, nous avons aussi compté pour chaque algorithme le nombre de solutions trouvées qui appartiennent au front Pareto optimal. Du fait de la taille des instances du problème considéré, nous n'avons pu calculer les fronts Pareto optimaux que pour les instances bi-objectifs de 250 et 500 objets.

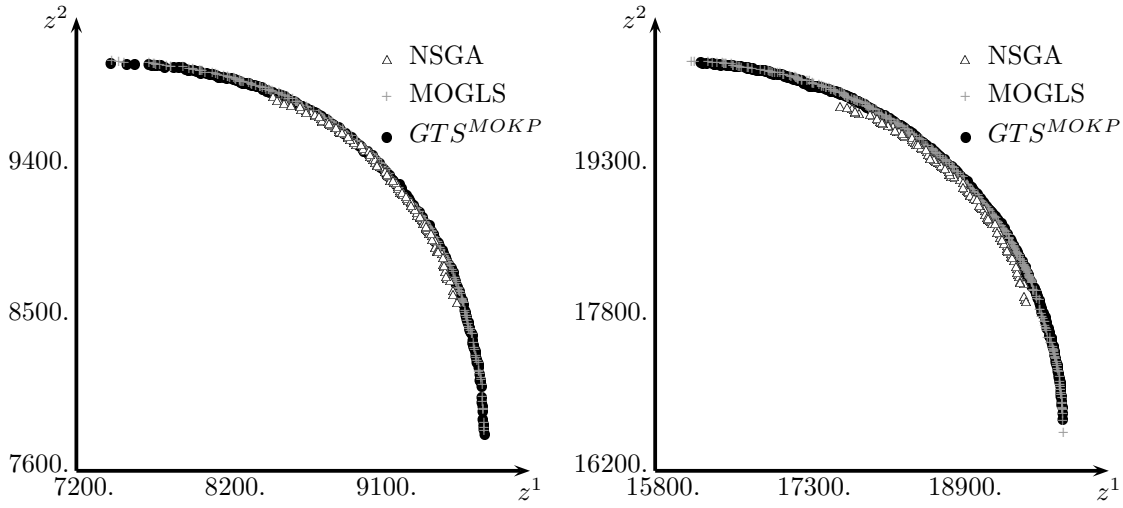
Une fois encore, GTS^{MOKP} obtient de meilleurs résultats, comparativement aux autres algorithmes *NSGA* et *MOGLS*. En effet, pour la première instance (250 objets), 9.33% des solutions trouvées par GTS^{MOKP} sont des éléments du front Pareto alors que ce taux n'est respectivement que de 0.18% pour *NSGA* et 6.51% pour *MOGLS*. Pour la seconde instance (500 objets), ce taux est respectivement de 0.35% pour GTS^{MOKP}, 0.07% pour *MOGLS* et 0% pour *NSGA*.

Pour les instances bi-objectifs, il est possible de représenter graphiquement les surfaces de compromis procurées par les différents algorithmes. La figure 5.2 montre ces surfaces pour chacune des trois instances bi-objectifs. Cette autre représentation vient confirmer les résultats chiffrés obtenus précédemment. Sur l'instance comportant 250 objets, les trois surfaces de compromis sont quasiment confondues, il est très difficile de départager les trois algorithmes. Pour les autres instances, il apparaît un peu plus clairement que la surface de compromis de *NSGA* est plus étriquée que celles de *MOGLS* et GTS^{MOKP}. De plus, nous observons que la surface de compromis de GTS^{MOKP} est légèrement plus étendue que celle calculée par *MOGLS*.

En outre, nous avons observé, dans les autres expérimentations que nous avons effectuées, où le nombre de générations pour GTS^{MOKP} est fixé à la même valeur que celui de *MOGLS*, que les résultats obtenus par GTS^{MOKP} sont encore meilleurs que ceux présentés dans ce document.

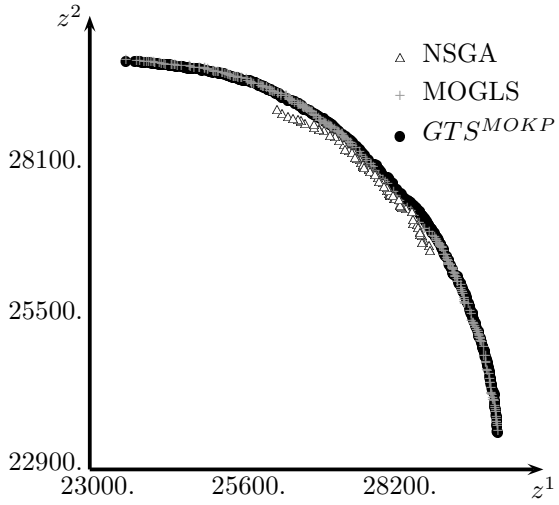
5.4 Conclusion

Dans ce chapitre, nous avons présenté GTS^{MOKP}, un algorithme génétique avec recherche Tabou très performant pour résoudre le problèmes du sac à dos multiobjectif multidimensionnel en 0-1 (MOKP01). Notre algorithme GTS^{MOKP} combine les concepts généraux des méthodes évolutionnaires avec les propriétés locales de la recherche Tabou, conduisant à un meilleur compromis entre exploitation et exploration. Les performances de GTS^{MOKP} ont été validées sur un ensemble de neuf instances publiées et comparées avec *NSGA* et *MOGLS*, deux algorithmes de l'état de l'art. Les résultats expérimentaux ont montré que GTS^{MOKP}, avec moins d'effort, obtient de meilleurs résultats que les autres algorithmes en compétition sur les instances testées. Nous avons ainsi mis en évidence l'importance de l'intensification. En effet, avec des méthodes de diversification équivalentes, l'algorithme possédant une meilleure méthode d'intensification obtient de meilleurs résultats. Ainsi, diversification et intensification sont deux concepts indissociables jouant un rôle majeur dans la résolution des problèmes multiobjectifs. Remarquons, que pour GTS^{MOKP} nous avons choisi *TS* et non pas *TS+HW*, ayant pourtant obtenu de meilleurs résultats avec ce dernier (cf. Section 4.4.3). En effet, la différence entre *TS* et *TS+HW* tient essentiellement



Instance avec 250 objets

Instance avec 500 objets



Instance avec 750 objets

FIG. 5.2 – *Surfaces de compromis procurées par les trois algorithmes sur trois instances du problème du sac à dos.*

à la manière de diversifier. En intégrant *TS+HW* dans un algorithme évolutionnaire, nous obtiendrions un algorithme incorporant deux opérateurs différents de diversification. Ces deux opérateurs se gêneraient mutuellement et, en définitive, la recherche s'en trouverait dégradée. Pour qu'un algorithme évolutionnaire puisse profiter du mécanisme de diversification de *TS+HW*, il faudrait extraire cet opérateur de *TS+HW* puis l'incorporer de manière adéquate au processus de sélection de l'algorithme évolutionnaire.

Les opérateurs génétique et l'opérateur de recherche Tabou utilisés dans GTS^{MOKP} sont implémentés de manière naturelle en se basant sur les principaux concepts de ces deux paradigmes. Pour cette raison, nous sommes fortement convaincus que beaucoup d'améliorations sont possibles. Par exemple, sur la partie génétique, d'autres stratégies de sélection peuvent être expérimentées, et des opérateurs de croisement spécifiques au problème peuvent être développés. Pour l'opérateur TS, nous pensons que les techniques développées pour le problème de sac à dos multidimensionnel classique, comme celles présentées dans [Vasquez and Hao, 2001], peuvent apporter beaucoup à un problème du type MOKP.

Troisième partie

Problèmes multiobjectifs continus sous contraintes

Chapitre 6

Une approche hybride pour l'optimisation multiobjectif

Dans ce chapitre, nous présentons *PICPA*, un nouvel algorithme pour traiter les problèmes multiobjectifs continus sous contraintes. Cet algorithme combine des techniques de propagation de contraintes avec des concepts évolutionnaires. À la différence des algorithmes évolutionnaires classiques, qui ne donnent que des solutions heuristiques, *PICPA* est capable de calculer des bornes du front Pareto optimal, tout en produisant des solutions approchées très précises. Une grande partie de ce chapitre est publiée dans [Barichard and Hao, 2003c].

Sommaire

6.1 Introduction

Dans cette partie de la thèse, nous nous intéressons plus particulièrement aux problèmes d'optimisation continus multiobjectifs. Les variables de décisions étaient jusqu'ici des variables discrètes, maintenant elles seront continues. Le cadre de l'optimisation des problèmes continus est plus général que celui des problèmes discrets, et les méthodes de résolution sont souvent différentes. Un problème d'optimisation continu multiobjectif sous contraintes peut être défini comme suit :

$$\begin{cases} \min & f_i(\overrightarrow{x}) & i = 1, ..., m \\ \text{t.q.} & C_l(\overrightarrow{x}) \geq 0 & l = 1, ..., q \\ & \overrightarrow{x} \in \mathbb{R}^n \end{cases}$$

où n représente le nombre de variables, $\overrightarrow{x}$ un vecteur de décision, m le nombre d'objectifs (critères) et q le nombre de contraintes du problème.

Comme nous l'avons vu au chapitre 2, beaucoup de méthodes évolutionnaires ont été développés pour résoudre les problèmes d'optimisation multiobjectif en général. Dans le cadre des problèmes d'optimisation continus multiobjectifs, certaines méthodes, comme `NSGA-IIc` de Deb [Deb and Goel, 2001], se sont révélées les plus adaptées et les plus efficaces.

Malheureusement, lorsque le nombre de contraintes à satisfaire augmente, les algorithmes évolutionnaires ont souvent des difficultés pour trouver des solutions réalisables de bonne qualité. En effet, bien que plusieurs méthodes spécifiques aient été introduites (cf. [Michalewicz and Schoenauer, 1996]), la plupart des algorithmes évolutionnaires agrègent les contraintes dans les fonctions objectifs en affectant à chacune un facteur de pénalité. Celui-ci peut être déterminé de manière statique, dynamique ou adaptative (cf. [Hamida and Schoenauer, 2000]). Mais ces pénalités sont agrégées, et il n'est plus possible d'identifier précisément une contrainte en particulier. Ainsi, plus le nombre de contraintes augmente et plus il est difficile (voire impossible) d'en différencier certaines. Dans ce cas, comment concentrer la recherche sur une contrainte en particulier ? Comment différencier les contraintes des objectifs ? Il est clair que le traitement des contraintes nécessite une attention particulière. De même, lorsque la zone réalisable dans l'espace des objectifs est non connexe, les algorithmes évolutionnaires convergent difficilement vers l'intégralité du front Pareto optimal. En effet, les algorithmes évolutionnaires ont la faculté de pouvoir oublier certaines zones de l'espace de recherche considérées comme peu prometteuses. Cette faculté est très appréciable pour permettre à la recherche de s'intensifier dans d'autres zones, seulement lorsque le choix de considérer la zone comme peu prometteuse est erroné, la zone considérée est souvent oubliée, même si elle contenait des solutions. Par exemple, lorsque l'espace de recherche est non connexe et que certaines composantes connexes sont très petites, l'algorithme peut manquer des solutions dans ces petites zones. De plus, ces algorithmes ne donnent aucune borne du front Pareto optimal, et ne peuvent pas estimer l'erreur entre les solutions approchées fournies et le front Pareto optimal.

Dans ce chapitre, nous présentons `PICPA` (*Population and Interval Constraint Propagation Algorithm*), un algorithme capable de produire des solutions approchées de très bonne qualité tout en calculant des bornes garanties du front Pareto optimal cherché. Ces

bornes nous permettent d'apprécier si les solutions trouvées sont proches ou éloignées du front Pareto. `PICPA` combine des techniques de propagation de contraintes sur intervalles : "Interval Constraint Propagation" (*ICP*) [Cleary, 1987; Davis, 1987] avec des concepts évolutionnaires (population et processus de sélection). Les expérimentations menées avec `PICPA` sur des problèmes test de la littérature ont montré l'efficacité de la méthode.

6.2 Analyse par intervalles et propagation des contraintes

Dans cette section, nous expliquons brièvement les concepts de base des mécanismes de propagation de contraintes modélisées par des intervalles (*Interval Constraint Propagation*, ICP). Les algorithmes d'ICP combinent le calcul par intervalles [Moore, 1979] et la propagation des contraintes [Mackworth, 1977] pour résoudre des systèmes d'équations et d'inéquations non linéaires. Les premiers algorithmes d'ICP furent introduits par Cleary [Cleary, 1987] et Davis [Davis, 1987].

6.2.1 Analyse par intervalle et contraction

Chaque variable du problème est représentée par un intervalle, et liée aux autres variables par une ou plusieurs contraintes. Il est clair que le domaine de la contrainte contenant toutes les affectations réalisables des variables induites est un sous-ensemble du produit cartésien de ces dites variables. Grâce aux techniques d'analyse par intervalle, il est possible, dans certains cas, de répercuter sur les domaines des variables certaines informations déduites des contraintes. Ces informations permettent de retirer du domaine des variables des valeurs ne participant à aucune solution, car violant une contrainte (c.à.d. l'affectation correspondante des variables n'est pas incluse dans le domaine de la contrainte).

Suite aux travaux de Moore [Moore, 1979], des opérateurs de réduction (ou contraction) ont été formalisés. En appliquant ces opérateurs sur une contrainte (ou fonction) il est possible de réduire le domaine d'une variable en retirant des valeurs inconsistantes, c.à.d ne participant à aucune solution du problème. Cette contraction n'affecte en rien les solutions du problème, en effet : *une réduction garantie d'intervalles par une fonction ou contrainte peut être calculée grâce à des opérateurs de réduction (ou contraction)*

Un exemple de contraction d'intervalle, en utilisant la contrainte *cubique*, est donné à la sous-section suivante.

De manière générale, pour chaque *contrainte primitive* (ou élémentaire), il existe un opérateur de réduction permettant de contracter chacun des domaines des variables. Pour plus d'informations sur ces opérateurs de contraction, le lecteur est invité à consulter le livre de L. Jaulin [Jaulin *et al.*, 2001].

6.2.2 Exemple de contraction avec la contrainte cubique

Considérons que les deux variables x et y appartiennent à des domaines dont toutes les valeurs sont à priori valides.

$$x \in [x^-, x^+] = [-2, 2]$$
$$y \in [y^-, y^+] = [-8, 9]$$

Considérons maintenant la contrainte : $y = x^3$. Il résulte de cette définition que la contrainte *cubique* est un sous ensemble de $\mathbb{R}^2$:

$$cubique = \{(x, y) \in \mathbb{R}^2 \mid y = x^3\}$$

La contrainte cubique est une contrainte binaire puisqu'elle prend en compte deux variables x et y. Dans notre exemple, la contrainte cubique peut être utilisée pour *enlever des valeurs inconsistantes* du domaine de y. En effet, nous observons que : $\forall x \in [-2, 2], \quad x^3 \leq 8$. Donc, toutes les valeurs du domaine de y plus grandes que 8 peuvent être retirées, car elles ne participent à aucune solution (cf. la zone hachurée de la figure 6.1).

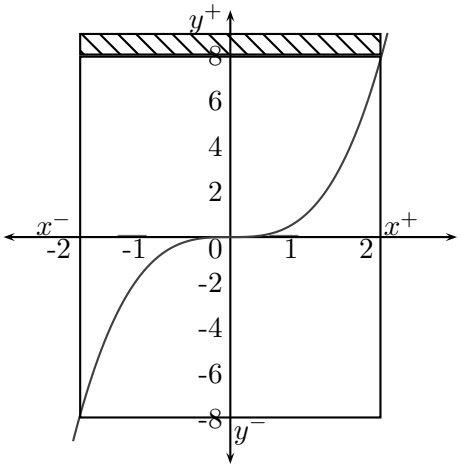


FIG. 6.1 – *Exemple de contraction sur la contrainte cubique.*

De manière plus formelle, la contrainte cubique permet de réduire le domaine de x et de y grâce aux opérateurs de contraction suivants :

$$\begin{cases} [x^-, x^+] & \longleftarrow & [x^-, x^+] \bigcap (\sqrt[3]{[y^-, y^+]}) \\ [y^-, y^+] & \longleftarrow & [y^-, y^+] \bigcap ([x^-, x^+]^3) \end{cases}$$

$$\begin{array}{lrcl} \text{où} & \sqrt[3]{[y^-, y^+]} & = & [\sqrt[3]{y^-}, \sqrt[3]{y^+}] \\ \text{et} & [x^-, x^+]^3 & = & [(x^-)^3, (x^+)^3] \end{array}$$

Cet opérateur de réduction (ou contraction) peut ensuite être réappliqué à chaque modification du domaine d'une des deux variables. En effet, un autre opérateur de contraction peut, en utilisant une autre contrainte du problème, être amené à réduire un domaine d'une de nos deux variables.

6.2.3 Atteindre un point fixe

Pour un ensemble de contraintes, l'application itérative de procédures de contraction (issues du calcul par intervalles) sur les contraintes conduit à l'obtention d'un état où plus aucun domaine de variables ne peut être réduit. Cet état est le point fixe de cet ensemble de contraintes. Notons que ce point fixe ne constitue pas une solution car les variables ne sont pas encore instanciées. Pour obtenir une solution, il est nécessaire d'utiliser en plus des méthodes d'exploration de l'espace de recherche.

Dans les approches de programmation par contraintes, plusieurs algorithmes de point fixe ont été développés. Nous pouvons citer les différentes versions d'algorithmes d'arc consistance pour les problèmes combinatoires [Mackworth, 1977; Mohr and Henderson, 1986], et les algorithmes d'ICP pour les problèmes continus [Benhamou *et al.*, 1999]. Bien que les algorithmes d'ICP soient principalement utilisés pour des problèmes à variables continues, ils peuvent être appliqués aussi sur des problèmes combinatoires.

6.2.4 Discussion

Un algorithme d'ICP est une procédure polynomiale qui réduit (contracte) le domaine des variables du problème. Cette famille d'algorithmes réduit l'espace de recherche en retirant des valeurs inconsistantes du domaine des variables par rapport à chaque contrainte du problème. À aucun moment, une solution du problème n'est enlevée. Cependant, l'application d'un algorithme d'ICP conduit seulement à l'obtention d'une approximation de la solution. Pour augmenter la précision de cette approximation, il est nécessaire de découper le domaine des variables et d'appliquer des algorithmes d'ICP sur les différentes alternatives obtenues. En itérant ce procédé, nous augmentons la précision, mais nous tendons vers un nombre exponentiel de découpes, et donc un très long temps de calcul. En pratique, cette approche n'est pas applicable aux problèmes ayant un grand nombre de variables.

Pour les problèmes d'optimisation multiobjectifs sous contraintes, il faut satisfaire un ensemble de contraintes, tout en optimisant plusieurs fonctions objectifs. En conséquence, même avec un petit nombre de variables et d'objectifs, le problème ne peut pas être traité simplement avec un algorithme classique de découpe. En effet, nous savons que l'optimisation d'un problème multiobjectif est souvent plus difficile que l'optimisation d'un problème mono-objectif, car la solution optimale ne se réduit plus à un seul point, mais à un ensemble de points non dominés.

6.3 Un algorithme combinant population et propagation de contraintes : *PICPA*

Le concept de population est très approprié dans un contexte multiobjectif. En effet, comme le front Pareto optimal est, dans la plupart des cas, un ensemble de solutions, chaque individu de la population peut espérer devenir une solution particulière du front Pareto optimal. En conséquence, la population dans son intégralité devient une approximation de l'ensemble Pareto optimal cherché.

Plusieurs algorithmes d'optimisation fondés sur le concept de population ont ainsi été développés. Nous pouvons citer, parmi d'autres, *NSGA* [Srinivas and Deb, 1994], *SPEA* [Zitzler and Thiele, 1999], *M-PAES* [Corne and Knowles, 2000]. Ces algorithmes ont montré leur efficacité sur un grand nombre de problèmes, et la notion de population y contribue largement. C'est pourquoi nous utilisons aussi cette notion de population, mais d'une façon différente.

Notre méthode *PICPA* est d'un concept original, car elle est à notre connaissance la seule intégrant dans un seul algorithme :

- Une représentation de chaque variable par un intervalle, permettant ainsi l'application des algorithmes d'ICP vus à la Section 6.2.

- Une caractérisation des individus par des boîtes, correspondant au produit cartésien des domaines des variables. À chaque individu est associée une partie de l'espace de recherche, et l'ensemble des individus (la population) encadre de manière garantie les solutions optimales.

- Un processus d'instanciation [1] pour associer à chaque boîte un point réalisable, permettant de déterminer des solutions approchées du front Pareto cherché.

- Un mécanisme de sélection utilisant la PS-dominance. Cet opérateur de sélection permet de conserver la garantie au fur et à mesure des générations. Ainsi, à chaque instant de la résolution, la population encadre le front Pareto optimal.

Dans cette section, nous présentons l'algorithme *PICPA* et décrivons plus précisément les concepts qui le composent.

6.3.1 Représentation de l'espace de recherche et des objectifs

Dans la plupart des algorithmes fondés sur le principe de population, un individu ou une configuration est un vecteur de décision, chaque variable étant instanciée par une valeur de $\mathbb{R}$. Avec cette représentation, chaque individu correspond à un point particulier dans l'espace des objectifs.

Dans notre approche, chaque individu est aussi un vecteur, mais chaque variable est maintenant représentée par un *intervalle*, au lieu d'une valeur simple. En conséquence, chaque individu de la population correspond maintenant à une *boîte* $[\overrightarrow{x}]$ de $\mathbb{R}^m$ (m étant le nombre d'objectifs du problème) dans l'espace des objectifs.

Considérons un problème à deux variables x_1, x_2 et deux objectifs f_1, f_2. Une population de trois individus pourrait être représentée comme à la figure 6.2.

Dans cette représentation, la population dans son intégralité décrit un sous-pavage régulier (i.e. une union de boîtes sans chevauchement) de $\mathbb{R}^m$. Il en résulte que les boîtes correspondantes de $\mathbb{R}^n$ peuvent, quant à elles, se chevaucher (cf. figure 6.2).

1. Le terme *instanciation* est communément employé dans la communauté Intelligence Artificielle pour désigner un processus affectant à chaque variable une valeur appartenant à son domaine de définition.

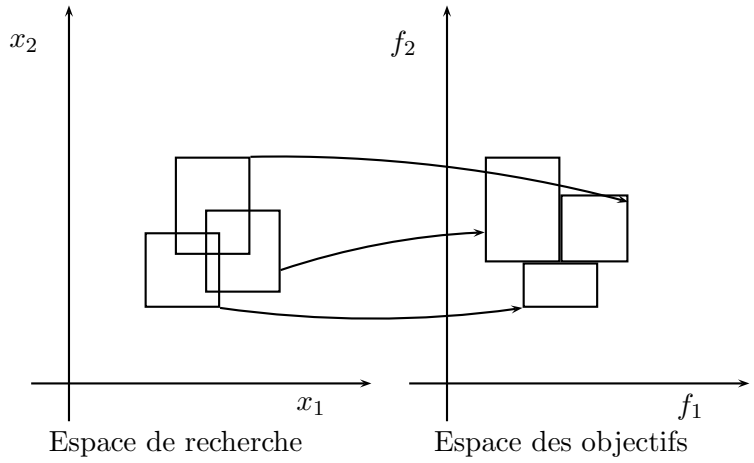


FIG. 6.2 – *Exemple de représentation avec des intervalles.*

6.3.2 Processus d'instanciation

Pour le moment, nous n'avons considéré que des boîtes. Cependant, nous cherchons aussi à trouver des solutions approchées du front Pareto. Pour calculer ces solutions, nous déclenchons un processus d'instanciation, qui va tenter de déterminer un point réalisable pour chaque $[\overrightarrow{x}]$. Plus précisément, une valeur particulière satisfaisant les contraintes, tout en restant incluse dans la boîte, sera recherchée, et ceci pour tous les individus de la population. Dans *PICPA*, cette recherche est effectuée naïvement, en essayant d'étendre une instanciation partielle à une instanciation globale (recherche exhaustive). Comme cette étape peut requérir un temps de calcul considérable, nous consacrons un *effort de recherche* limité à cette recherche. Cet *effort de recherche* est un paramètre qui permet d'interrompre le processus si aucun point réalisable n'a encore été trouvé. Nous utilisons la valeur de l'effort de recherche pour borner son nombre de retours en arrière lorsque l'instanciation a échoué.

L'algorithme 10 décrit le processus d'instanciation. À chaque appel de la procédure, une phase d'affectation est appelée. Cette phase consiste à étendre une instanciation partielle en partant de la variable ayant le plus grand domaine. Une fois la première variable sélectionnée, nous lui affectons comme valeur le milieu de l'intervalle de son domaine, puis le mécanisme habituel de propagation propage cette réduction aux autres variables. Nous sélectionnons ensuite la variable non instanciée ayant le plus grand domaine, puis nous poursuivons notre procédure d'instanciation en repartant de la variable sélectionnée. Cette phase s'arrête dès qu'une affectation viole une contrainte, ou que l'affectation de toutes les variables a été effectuée. Si la phase d'affectation a échoué, la procédure d'instanciation est rappelée récursivement sur deux boîtes plus petites qui ont été réduites par des algorithmes d'ICP (cf. Section 6.2). La procédure d'instanciation s'arrête dès que la taille de la plus grande variable de I est inférieure à la taille d'arrêt. Cette taille d'arrêt est calculée au préalable à partir de la taille de la plus grande arête de la boîte initiale et du paramètre

d'*effort de recherche*. L'appel à la procédure d'instanciation est effectué de la manière suivante :

instanciation(I, ((1−*effort de recherche*) × *taille de la plus grande variable de* I))

Algorithme 10 Procédure d'instanciation de *PICPA*.

```
Procédure instanciation(I,taille_Arrêt)
    taille ⟵ taille de la plus grande variable de I
    Si taille < taille_Arrêt alors
        Essayer d'instancier les variables de I
        Si I non instancié alors
            Couper I pour obtenir I1 et I2
            ∀x ∈ {j | j ∈ {ICP(I1),ICP(I2)}, j est localement consistant},
                                    exécuter instanciation(x, taille_Arrêt)
        FinSi
    FinSi
FinProcédure
```

Nous n'autorisons donc qu'un faible nombre de tentatives pour découvrir un point réalisable. Le nombre de tentatives d'instanciation croît proportionnellement à la valeur du paramètre d'effort de recherche. Ainsi, après ce stade, certains individus de la population contiennent un point complètement instancié, c'est-à-dire un vecteur réel $\overrightarrow{x}$, alors que d'autres ne contiennent aucun point. Notons que le résultat de chaque instanciation (chaque point réalisable trouvé) est stocké dans une structure de données séparée, et que les individus de la population ne sont pas modifiés.

6.3.3 Une relation de dominance "point-ensemble"

Nous introduisons une relation de dominance entre un point et un ensemble (PS-dominance). La PS-dominance est une extension de la relation de dominance classique (cf. Définition 7 page 11) :

Définition 20 (PS-dominance) *Soit un vecteur* $\mathbf{u}$ *et un ensemble de vecteurs* $\{\mathbf{v}\}$, $|\{\mathbf{v}\}| > 0$*:*

$\mathbf{u} =_{ps} \{\mathbf{v}\}$ *(*$\mathbf{u}$ *PS-égale* $\{\mathbf{v}\}$*), ssi* $\forall \mathbf{v} \in \{\mathbf{v}\} : \mathbf{u} = \mathbf{v}$

$\mathbf{u} \prec_{ps} \{\mathbf{v}\}$ *(*$\mathbf{u}$ *PS-domine* $\{\mathbf{v}\}$*), ssi* $\forall \mathbf{v} \in \{\mathbf{v}\} : \mathbf{u} \prec \mathbf{v}$

$\mathbf{u} \succ_{ps} \{\mathbf{v}\}$ *(*$\mathbf{u}$ *est PS-dominé par* $\{\mathbf{v}\}$*), ssi* $\forall \mathbf{v} \in \{\mathbf{v}\} : \mathbf{u} \succ \mathbf{v}$

$\mathbf{u} \sim_{ps} \{\mathbf{v}\}$ *(*$\mathbf{u}$ *est incomparable (PS-non dominé) avec* $\{\mathbf{v}\}$*), dans les autres cas*

La complexité de la PS-dominance est en $\mathcal{O}(m \times |\{\mathbf{v}\}|)$ car nous devons tester $\mathbf{u}$ avec chaque élément de $\{\mathbf{v}\}$.

Pour la suite, nous utilisons la notation $[v]$ pour désigner un intervalle de $\mathbb{R}$ et $\overrightarrow{[v]}$ pour désigner un *vecteur d'intervalles* de $\mathbb{R}^m$. De plus, un *vecteur d'intervalles* de $\mathbb{R}^m$ peut aussi être appelé une boîte de $\mathbb{R}^m$. Il en résulte que nous pouvons appliquer la PS-dominance entre n'importe quel vecteur $\overrightarrow{u}$ de $\mathbb{R}^m$ et n'importe quelle boîte $\overrightarrow{[v]}$ de $\mathbb{R}^m$ (cf. figure 6.3). Dans ce cas particulier, la complexité de la PS-dominance est en $\mathcal{O}(m)$. En effet, nous avons besoin de tester la dominance uniquement entre $\overrightarrow{u}$ et le point situé au coin inférieur gauche de $\overrightarrow{[v]}$.

Considérons une boîte $\overrightarrow{[v]}$ de $\mathbb{R}^2$ et $\overrightarrow{u}$ un point de $\mathbb{R}^2$. Pour un problème de minimisation, nous pouvons illustrer la PS-dominance par les cas donnés à la figure 6.3.

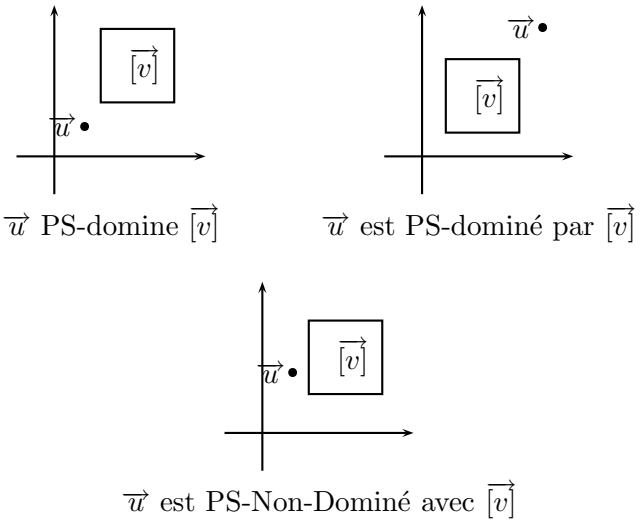


FIG. 6.3 – *Exemples de cas de PS-dominance.*

6.3.4 Sélection Pareto de boîtes

Considérons un ensemble d'individus représentés par des boîtes (cf. Section 6.3.1). Nous sommes assurés que toutes les configurations réalisables sont contenues dans le sous-pavage décrit par la population. En conséquence, le front Pareto optimal se trouve aussi inclus dans la population.

Pour enlever des individus qui ne contiennent aucune solution de l'ensemble Pareto optimal, nous appliquons la procédure de sélection Pareto suivante :

1. Essayer d'instancier les individus de la population en ne passant qu'un temps limité pour la recherche de solution.

2. Appliquer la PS-dominance (cf. section 6.3.3) pour retirer tous les individus qui sont dominés par un autre individu instancié de la population.

Après ce traitement, nous avons une population réduite d'individus, et nous sommes assurés que l'union de ces boîtes (ou individus) contient toujours l'intégralité du front Pareto optimal. Nous pouvons donc énoncer le théorème suivant :

Théorème 1 *Toute population obtenue par applications successives d'algorithmes d'ICP et de phases de sélection utilisant la PS-dominance contient le front Pareto optimal.*

Nous donnons maintenant une justification informelle de ce théorème : soient $\mathcal{X}$ l'ensemble réalisable dans l'espace de décision, $\overrightarrow{y}$ une configuration du front Pareto optimal ($\overrightarrow{y}$ est dans l'espace de décision) et $\overrightarrow{[y]}$ une boîte contenant $\overrightarrow{y}$:

$$\begin{array}{l} \text{donc, } \nexists \overrightarrow{x} \in \mathcal{X} \text{ et } \overrightarrow{x} \prec \overrightarrow{y} \\ \text{donc, } \nexists \overrightarrow{x} \in \mathcal{X} \text{ et } \overrightarrow{x} \prec_{ps} \overrightarrow{[y]} \end{array}$$

En conséquence, $\overrightarrow{[y]}$ ne peut être PS-dominé et ne peut pas être enlevé de la population. Ainsi, $f(\overrightarrow{[y]})$ qui est l'image de $\overrightarrow{[y]}$ dans l'espace des objectifs ne peut être retirée.

La figure 6.4 montre un exemple de sélection Pareto avec la PS-dominance. Il est clair que les boîtes hachurées peuvent être enlevées, car elles sont PS-dominées par des points réalisables de l'espace des objectifs.

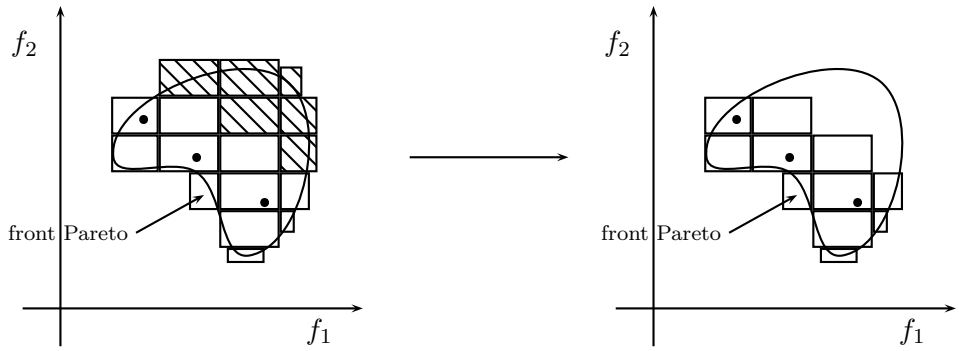


FIG. 6.4 – *Un exemple de sélection Pareto.*

6.3.5 La méthode *PICPA*

PICPA combine des mécanismes d'ICP (*Interval Constraint Propagation*) avec un processus de sélection Pareto (cf. Section 6.3.4) en un seul algorithme.

PICPA utilise une population de taille variable, sa taille maximale étant un paramètre à fixer. *PICPA* démarre avec un individu unique $\overrightarrow{[x]}$ dans lequel chaque variable x_i est initialisée par un intervalle (le domaine de définition de la variable). Nous supposons que chaque variable est réduite à l'intervalle minimum vérifiant la consistance locale. Si ce n'est pas le cas, une phase d'ICP est d'abord appliquée pour atteindre un point fixe. Il est clair que cet individu correspond à une boîte de $\mathbb{R}^m$.

Soit $\overrightarrow{[f]}$ cette boîte. Prenons un objectif f_i (par exemple, celui dont l'intervalle de définition est le plus grand) et réalisons une bissection dans son intervalle de définition $[f_i]$.

Cette bissection sur $\overrightarrow{[x]}$ conduit ainsi à l'obtention de deux nouveaux individus $\overrightarrow{[x']}$ et $\overrightarrow{[x'']}$. Un processus d'ICP est ensuite appliqué à chacun de ces deux nouveaux individus dans le but de réduire l'intervalle de chacune de leurs variables. Ces individus remplacent l'individu *parent* et correspondent ainsi à deux boîtes $\overrightarrow{[f']}$ et $\overrightarrow{[f'']}$ de $\mathbb{R}^m$. Ajoutons que si le domaine d'une variable est réduit à l'ensemble vide pendant une phase d'ICP, l'individu ainsi mis en évidence ne sera pas réinséré dans la population. Ce mécanisme de "bissection-réduction" est itéré jusqu'à ce que le nombre d'individus de la population atteigne la taille maximale. Les boîtes étant insérées dans une file au fur et à mesure des découpes, la prochaine boîte à découper sera en fait la première de la file. Lorsqu'une boîte est sélectionnée, il reste à choisir sur quel objectif nous allons procéder à la découpe. Pour cela nous choisissons l'objectif ayant le plus grand domaine de définition.

Une fois que la population a atteint sa taille maximale, un processus d'instanciation va être déclenché dans le but d'obtenir un point réalisable pour chaque $\overrightarrow{[x]}$ (cf. Section 6.3.2). Ensuite, nous pouvons appliquer le mécanisme de sélection Pareto (cf. section 6.3.4) pour éliminer les individus PS-dominés de la population. Comme la taille de population est réduite, nous pouvons itérer le processus de "bissection-réduction" vu précédemment pour ré-augmenter la taille de la population jusqu'à sa taille maximale.

L'algorithme *PICPA* s'arrête si l'une des conditions suivantes est rencontrée :

1. une population vide est rencontrée, dans ce cas, le problème n'est pas réalisable, en conséquence aucune solution ne peut être trouvée,

2. le processus de sélection Pareto ne peut enlever aucun individu de la population. Dans ce cas, les individus de la population constituent une borne des solutions optimales du problème. Chaque individu qui a été instancié avec succès donne en plus une solution approchée d'un point du front Pareto optimal.

Nous donnons maintenant le pseudo-code de *PICPA* dans l'algorithme 11.

Algorithme 11 Pseudo-code de l'algorithme *PICPA*.

Initialiser la population avec un unique individu localement consistant
`Tant que` $0 < |Population| < Max\_Population\_Size$ `faire`
    `Tant que` $0 < |Population| < Max\_Population\_Size$ `faire`
        - Sélectionner un individu (parent) et réaliser une bissection selon
          un des objectifs, conduisant à deux individus (enfants) distincts
        - Réduction des domaines des enfants (ICP)
        - Mise à jour de la population :
            (a) Retirer le père
            (b) Ajouter les enfants localement consistants
    `FinTantQue`
    - Instanciation potentielle de chaque individu
    - Processus de sélection Pareto (PS-dominance)
`FinTantQue`

Remarquons que dans l'algorithme 11, seulement deux paramètres sont nécessaires à *PICPA* (la taille maximale de la population et l'effort de recherche).

PICPA a plusieurs avantages en comparaison avec d'autres algorithmes basés sur le concept de population. Premièrement, il nécessite très peu de paramètres. Deuxièmement, il peut parfois répondre "Non" lorsque le problème est infaisable. Troisièmement, il calcule, en une seule exécution, des bornes et une approximation du front Pareto optimal.

PICPA traite des problèmes d'optimisation multiobjectifs sous contraintes. Pour que *PICPA* soit à même de résoudre le problème, les contraintes et les objectifs doivent être décrits de manière analytique. Ainsi, *PICPA* comprend les opérateurs suivants :

- les opérateurs binaires classiques ($+$, $-$, $\times$, $/$),
- des opérateurs trigonométriques (sin, cos, tan, arcsin, arccos),
- des opérateurs de puissances (^{2}, $\sqrt{}$, (x^n, $n \in \mathbb{N}$)),
- mais aussi les opérateurs exponentiel (exp), logarithme (log), valeur absolue ($|\ |$), minimum et maximum de deux réels (min, max).

6.3.6 Discussion

PICPA garantit que le front Pareto optimal est inclus dans la population résultat. Comme *PICPA* réalise principalement des bissections dans l'espace des objectifs, il est moins sensible à l'augmentation du nombre de variables. Partitionner l'espace des objectifs est une méthode déjà étudiée de plusieurs manières. Nous pouvons citer [Hamda *et al.*, 2002; Hughes, 2003] qui utilisent la représentation de Voronoi, mais aussi [Schütze *et al.*, 2003] qui procède par recouvrement grâce à des collections de boîtes. Mais *PICPA* est une méthode originale, car elle est la seule à notre connaissance à intégrer dans un seul algorithme les points suivants :

1. une limitation du nombre de découpes (bissections), grâce à l'utilisation d'une population de taille bornée,
2. une stratégie de découpe dans l'espace des objectifs garantissant de ne perdre aucune solution optimale,
3. l'application d'un mécanisme de sélection Pareto pour converger vers le front Pareto optimal.

L'effort de calcul requis par *PICPA* peut être réglé en changeant la taille de population. En effet, plus grande est la population et plus les solutions calculées seront précises. Il est clair que l'augmentation de la taille de la population augmente le temps de calcul. Mais cela nous garantit d'obtenir une meilleure approximation du front Pareto optimal.

6.4 Résultats expérimentaux

Dans cette section, nous donnons les résultats expérimentaux de *PICPA* sur plusieurs problèmes test bi-critères de la littérature. Cependant, pour montrer ses performances pratiques, nous comparons les résultats de *PICPA* avec ceux de *NSGA-IIc* [Deb and Goel, 2001] [2]. Notons que la version de *NSGA-IIc* utilisée ici donne de meilleurs résultats [3] que ceux présentés dans [Deb and Goel, 2001]. Pour cette série de tests, nous utilisons les paramètres suivants :

- Pour *NSGA-IIc*, nous utilisons les paramètres donnés dans [Deb *et al.*, 2001], c'est-à-dire, un *croisement binaire simulé* [Deb and Agrawal, 1995] avec $n_c = 20$ et un *opérateur de mutation polynomiale* avec $n_m = 20$. La probabilité de croisement est fixée à 0.9 et le taux de mutation à 0.15. La taille de la population, ainsi que le nombre de générations, sont fixés en fonction de la difficulté du problème testé.

- Pour *PICPA*, nous fixons empiriquement la taille de la population à 1000 et l'effort de recherche à 0.2. Cette valeur de l'effort de recherche correspond à 80% de la taille du plus grand domaine des variables (cf. Section 6.3.2).

Notons que les valeurs de ces paramètres amènent les deux algorithmes à des temps de calcul similaires, allant de quelques secondes à quelques minutes selon le problème, sur un PC (Bi-Pentium III 1 Ghz) sous Linux. *NSGA-IIc* étant un algorithme stochastique, nous l'exécutons dix fois et gardons la *meilleure* exécution pour nos comparaisons. Étant donnée la nature déterministe de *PICPA*, une seule exécution est nécessaire.

6.4.1 Le problème de Tanaka

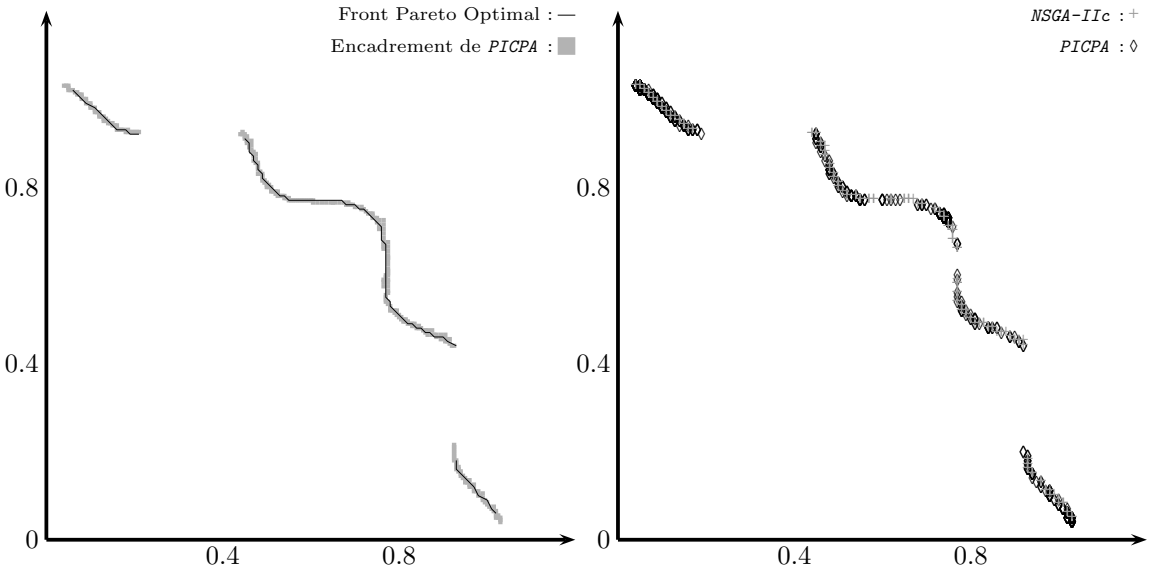
Nous utilisons d'abord un problème présenté par Tanaka [Tanaka, 1995]:

$$\text{TNK} \begin{cases} \text{Minimiser} & f_1(x) = x_1 \\ \text{Minimiser} & f_2(x) = x_2 \\ \text{t.q.} & c_1(x) \equiv x_1^2 + x_2^2 - 1 - 0.1\cos(16\arctan(\frac{x_1}{x_2})) \geq 0 \\ & c_2(x) \equiv (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5 \\ \text{et} & x_1, x_2 \in [0..\pi] \end{cases}$$

Ce problème comporte 2 variables et 2 contraintes. Le problème de Tanaka est largement utilisé par les chercheurs pour tester leur algorithme. Dans ce problème, la zone réalisable dans l'espace des objectifs est la même que dans l'espace de recherche. Pour les expérimentations sur TNK, nous utilisons une population de taille 150 et un nombre maximum de générations de 500 pour *NSGA-IIc*. Le front Pareto optimal, ainsi que l'encadrement donné par *PICPA*, sont présentés à la figure 6.5(a). La figure 6.5(b) montre les solutions approchées calculées par *PICPA* et *NSGA-IIc*.

2. Téléchargeable à l'adresse : `http://www.iitk.ac.in/kangal/soft.htm`

3. La version de *NSGA-IIc* téléchargée et utilisée ici est plus récente que la date de parution de l'article [Deb and Goel, 2001].



(a) Bornes du front Pareto optimal.

(b) Ensemble de solutions non dominées.

FIG. 6.5 – *Résultats expérimentaux pour le problème TNK.*

D'après ces figures, nous observons que premièrement les bornes de *PICPA* encadrent presque parfaitement le front Pareto optimal et que, deuxièmement, *PICPA* et *NSGA-IIc* calculent des solutions approchées comparables.

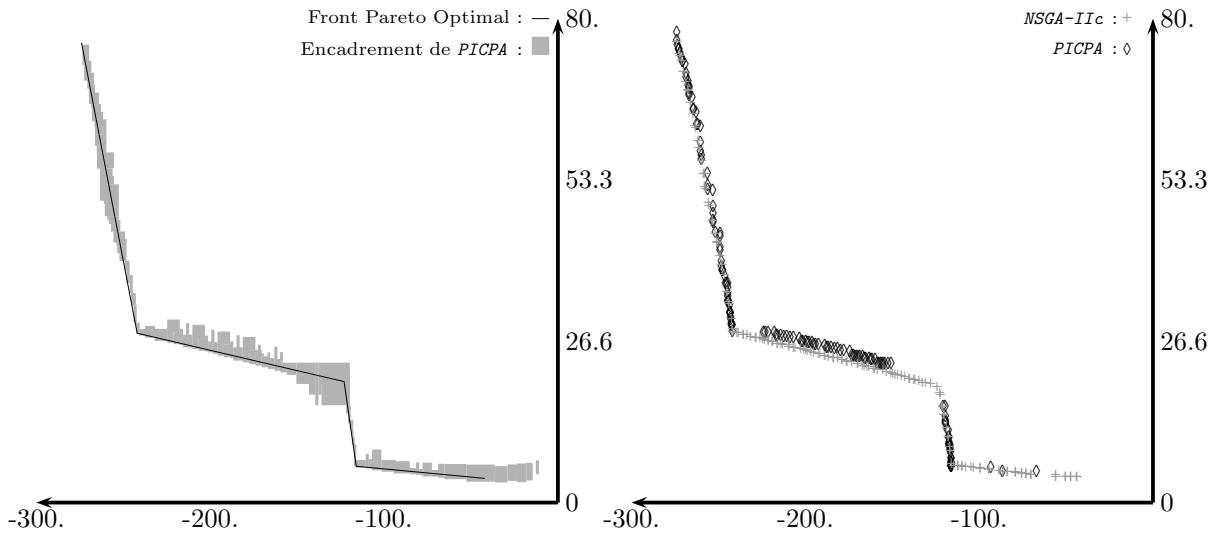
6.4.2 Le problème de Osyczka et Kundu

Ce problème à six variables est présenté par Osyczka et Kundu [Osyczka and Kundu, 1995] :

$$
\text{OSY}\begin{cases}
\text{Minimiser} & f_1(x) = -(25(x_1-2)^2+(x_2-2)^2+(x_3-1)^2+ \\
 & \qquad (x_4-4)^2+(x_5-1)^2) \\
\text{Minimiser} & f_2(x) = x_1^2+x_2^2+x_3^2+x_4^2+x_5^2+x_6^2 \\
\text{t.q.} & c_1(x) \equiv x_1+x_2-2 \geq 0 \\
 & c_2(x) \equiv 6-x_1-x_2 \geq 0 \\
 & c_3(x) \equiv 2-x_2+x_1 \geq 0 \\
 & c_4(x) \equiv 2-x_1+3x_2 \geq 0 \\
 & c_5(x) \equiv 4-(x_3-3)^2-x_4 \geq 0 \\
 & c_6(x) \equiv (x_5-3)^2+x_6-4 \geq 0 \\
\text{et} & x_1, x_2, x_6 \in [0..10] \\
 & x_3, x5, \in [1..5] \\
 & x_4, \in [0..6]
\end{cases}
$$

Ce problème comporte 6 variables et 6 contraintes. Nous utilisons ce problème car il comporte beaucoup d'opérateurs linéaires. En effet, la réduction de domaine sur des opérateurs linéaires est moins efficace que sur des contraintes non linéaires comme la puissance. Une modification sur le domaine de la contrainte peut souvent entraîner une mise à jour des domaines des variables concernées. Nous voulons observer le comportement de *PICPA* sur ce type de problème où a priori la contraction est moins efficace.

Pour ce problème, nous fixons la taille de la population de *NSGA-IIc* à 150, et son nombre de générations à 500.



(a) Bornes du front Pareto optimal.

(b) Ensemble de solutions non dominées.

FIG. 6.6 – *Résultats expérimentaux pour le problème OSY.*

D'après la figure 6.6(a), nous remarquons que les bornes de *PICPA* sont globalement très proches du front Pareto optimal. Certaines parties du front Pareto sont encadrées plus grossièrement à cause du pourcentage élevé de contraintes additives. La figure 6.6(b) montre que les qualités des deux ensembles de solutions non dominées trouvés par *PICPA* et *NSGA-IIc* sont très proches. Sur les zones un peu plus larges, *NSGA-IIc* trouve quelques points qui dominent ceux de *PICPA*, mais il est à noter que différentes exécutions de *NSGA-IIc* donnent des résultats de qualité variable. Notons de plus, que les résultats de la figure 6.6(b) correspondent à la meilleure approximation calculée par *NSGA-IIc*.

6.4.3 Des problèmes test avec contraintes (CTP)

Pour cette série de tests, nous utilisons deux problèmes présentés dans [Deb *et al.*, 2001] et [Deb, 2001]. Comme la difficulté de ces problèmes est réglable, nous utilisons une fonc-

tion $g()$ basée sur la fonction de Rastrigin. La fonction choisie pour ces expérimentations fait intervenir 5 variables :

$$\text{CTP} \begin{cases} \text{Minimiser} & f_1(x) & = & x_1 \\ \text{Minimiser} & f_2(x) & = & g(x) - f_1(x) \\ \text{t.q.} & g(x) & = & 41 + x_2^2 - 10\cos(4\pi x_2) + x_3^2 - 10\cos(4\pi x_3) + \\ & & & x_4^2 - 10\cos(4\pi x_4) + x_5^2 - 10\cos(4\pi x_5) \\ & C_1(x) & \equiv & \cos(\theta)(f_2(x) - e) - \sin(\theta) f_1(x) \geq \\ & & & a|\sin(b\pi(\sin(\theta)(f_2(x) - e) + \cos(\theta) f_1(x))^c)|^d \\ \text{et} & x_1 \in [0..1] & & \\ & x_{i,i>1}, \in [-5..5] & & \end{cases}$$

Ce générateur de problèmes est très récent. Il permet de générer des problèmes avec beaucoup de contraintes fortement non linéaires souvent difficiles à traiter par un algorithme d'optimisation.

Pour les expérimentations qui suivent, nous fixons la taille de la population pour *NSGA-IIc* à 300 et son nombre de générations à 1000. Les paramètres pour *PICPA* restent les mêmes que dans la section précédente.

CTP7

Notre première instance du générateur CTP comporte 5 variables et 1 contrainte. Les paramètres utilisés pour générer cette instance CTP7 sont les suivants :

$$\theta = -0.05\pi, \quad a = 40, \quad b = 5, \quad c = 1, \quad d = 6, \quad e = 0$$

L'espace de recherche réalisable, les zones Pareto optimales non connexes et les bornes trouvées par *PICPA* sont montrés à la figure 6.7(a). Nous constatons que les bornes calculées par *PICPA* entourent de manière très précise le front Pareto optimal. Sur ce problème plus difficile pour les algorithmes classiques, les mécanismes de réduction (contraction) utilisés par *PICPA* se sont montrés très efficaces. À la figure 6.7(b), nous observons que *PICPA* et *NSGA-IIc* donnent des solutions approchées comparables.

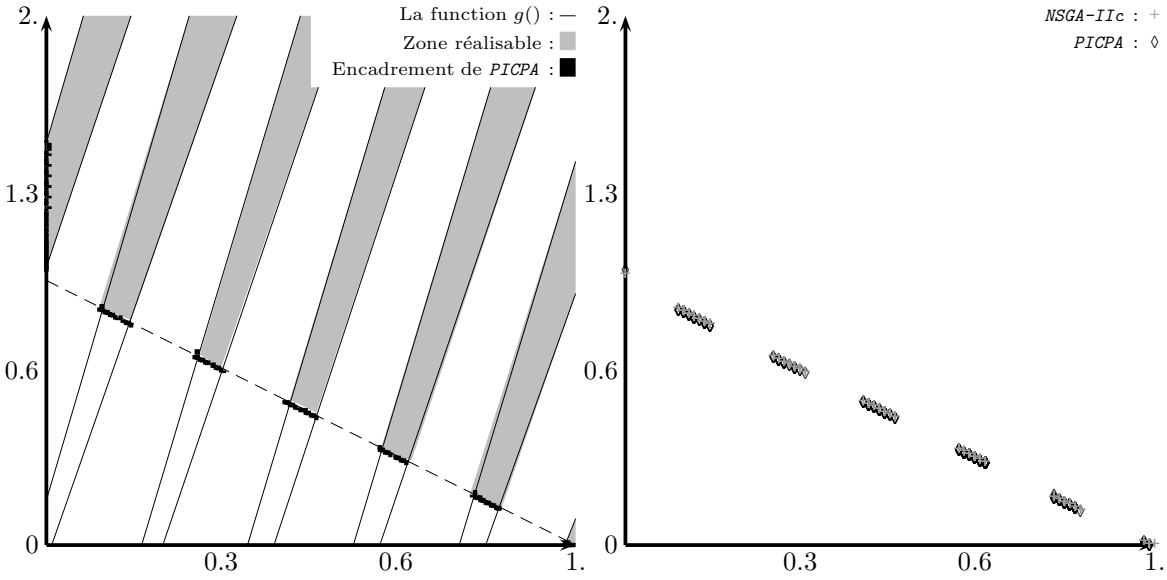
CTP8

Notre deuxième instance du générateur CTP comporte 5 variables et 2 contraintes. L'instance CTP8 générée est composée de plusieurs zones réalisables non connexes. Voici les paramètres utilisés pour générer les deux contraintes de l'instance :

$$C_1 \ : \ \theta = 0.1\pi, \quad a = 40, \quad b = 0.5, \quad c = 1, \quad d = 2, \quad e = -2$$

$$C_2 \ : \ \theta = -0.05\pi, \quad a = 40, \quad b = 2, \quad c = 1, \quad d = 6, \quad e = 0$$

L'espace de recherche réalisable, les zones Pareto optimales non connexes et les bornes trouvées par *PICPA* sont montrés à la figure 6.8(a). La figure 6.8(b) montre les ensembles de solutions non dominées trouvés par *NSGA-IIc* et *PICPA*.



(a) Bornes du front Pareto optimal.

(b) Ensemble de solutions non dominées.

FIG. 6.7 – *Résultats expérimentaux pour le problème CTP7.*

Sur la figure 6.8(a), nous observons qu'une fois encore l'encadrement du front Pareto calculé par *PICPA* est très précis. Sur ce problème, qui est également le plus difficile de ceux testés ici, *NSGA-IIc* et *PICPA* trouvent des solutions sur la plupart des composantes connexes du front Pareto optimal.

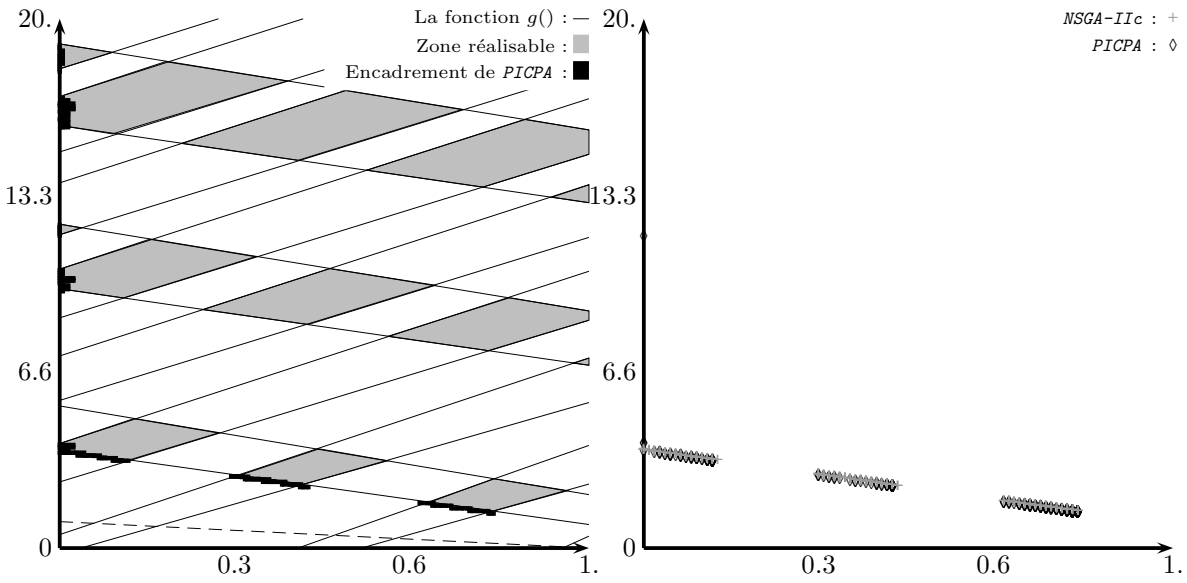
6.5 Conclusion

Dans ce chapitre, nous avons présenté *PICPA*, un nouvel algorithme pour résoudre les problèmes multiobjectifs sous contraintes. *PICPA* combine des mécanismes de propagation de contraintes et d'analyse par intervalles avec des concepts évolutionnaires (population et sélection). Ces différents concepts confèrent à *PICPA* plusieurs avantages :

- *PICPA* est complètement déterministe. Le fait de lancer l'algorithme une fois et d'obtenir toujours la même solution peut être un atout majeur pour certaines applications notamment industrielles : debuggage, SAV (relation concepteur et utilisateur).

- *PICPA* calcule des solutions approchées du front Pareto optimal, permettant au décideur d'arrêter son choix sur des configurations réalisables.

- *PICPA* détermine un encadrement du front Pareto cherché. Cet aspect est très important pour un décideur qui veut pouvoir évaluer l'erreur sur les solutions fournies.



(a) Bornes du front Pareto optimal.

(b) Ensemble de solutions non dominées.

FIG. 6.8 – *Résultats expérimentaux pour le problème CTP8.*

Les résultats expérimentaux de *PICPA* sur des problèmes test de la littérature ont permis d'apprécier son bon comportement. L'encadrement fourni est déjà très représentatif du front Pareto optimal, et les solutions approchées calculées sont de bonne qualité. De plus, la PS-dominance, qui compare un point à un ensemble de points, a aussi été présentée.

Ce travail est un premier pas dans l'hybridation de méthodes exactes et approchées. Ainsi, l'algorithme *PICPA* peut être amélioré sur plusieurs points :

- Premièrement, sur le processus d'instanciation qui, dans cette version de l'algorithme, est implémenté de manière naïve. Nous pensons que la recherche locale peut apporter beaucoup au mécanisme d'instanciation.
- Deuxièmement, sur le mécanisme de sélection qui, par une extension du principe de PS-dominance à des parties de boîtes, doit pouvoir éliminer encore plus de zones ne contenant pas de solutions optimales.

Au chapitre 7, nous approfondissons ces deux points et développons une nouvelle méthode conservant le même esprit, mais repoussant certaines limites de *PICPA*.

Chapitre 7

Un approfondissement des mécanismes de sélection et d'instanciation

Dans ce chapitre, nous présentons *PICPA-II*, une amélioration de notre précédent algorithme hybride *PICPA*. Nous étudions plus précisément le mécanisme de sélection et l'opérateur d'instanciation que nous améliorons à l'aide d'une recherche locale. Nous montrons que *PICPA-II* améliore tous les résultats obtenus par *PICPA*. En contrepartie, il perd le caractère déterministe de *PICPA*. Une partie de ce chapitre a été soumis pour publication [Barichard and Hao, 2003b].

Sommaire

7.1 Introduction

Dans ce chapitre, nous approfondissons les travaux présentés au chapitre 6. *PICPA* est un algorithme évolutionnaire original grâce à son aptitude à calculer des bornes du front Pareto optimal. Malheureusement, même s'il est moins sensible à l'augmentation du nombre de variables qu'un algorithme exact, il reste plus lent que certains algorithmes évolutionnaires. Notre principale motivation dans ce chapitre est donc d'améliorer l'efficacité de *PICPA*. Pour cela, nous allons nous concentrer sur deux points : *la sélection* et *l'instanciation*[1].

Nous allons, dans un premier temps, approfondir l'opérateur de sélection introduit à la Section 6.3.4. Nous présentons un nouvel opérateur de sélection permettant d'invalider certaines parties de boîtes non PS-dominées avec un point instancié.

Dans un deuxième temps, nous allons étudier en détail le mécanisme d'instanciation. Nous pensons que la recherche locale peut s'avérer une très bonne alternative au mécanisme développé pour *PICPA*. Pour construire un bon opérateur d'instanciation, nous étudions plusieurs choix d'algorithmes possibles et nous les soumettons à plusieurs expérimentations. Nous développons aussi un mécanisme original de *guidage* de la recherche locale grâce à la propagation de contraintes.

Enfin, nous décrivons l'algorithme *PICPA-II*, qui incorpore les opérateurs améliorés présentés dans ce chapitre, puis nous effectuons des expérimentations sur des jeux d'essais de la littérature.

7.2 Une sélection Pareto améliorée

Comme nous l'avons vu précédemment, le processus de sélection Pareto peut retirer des individus ne contenant pas de solution optimale. La figure 7.1 montre un exemple de sélection Pareto déjà évoqué dans le chapitre précédent (cf. Section 6.3.4). Dans cet exemple, les boîtes hachurées sont enlevées car elles sont PS-dominées par des points réalisables de l'espace des objectifs.

La relation de PS-dominance s'applique aux boîtes de la population, mais il est intéressant de noter que ces boîtes peuvent être invalidées (comme pour l'algorithme *PICPA*) ou contractées, si elles ne sont pas déclarées inconsistantes. En effet, dans chaque boîte, il est parfois possible de trouver des zones qui sont PS-dominées par des points instanciés. Il est donc possible de retirer ces zones et de réduire l'individu concerné.

Nous donnons à la figure 7.2 un exemple de processus d'une sélection Pareto améliorée. Cet exemple fait suite à la figure 7.1 et permet de mieux apprécier les effets de l'amélioration apportée. Nous observons que les zones hachurées peuvent être retirées car elles sont PS-dominées par des points réalisables de l'espace des objectifs.

Notons qu'il est possible de retirer des zones des boîtes instanciées de la même manière que pour les autres boîtes. Si nous enlevions ces zones, cela entraînerait la création de nouveaux individus dans la population. En effet, si nous réduisons une boîte instanciée, le

1. Nous rappelons qu'un opérateur d'instanciation consiste à affecter à chacune des variables, une valeur de leur domaine satisfaisant toutes les contraintes du problème.

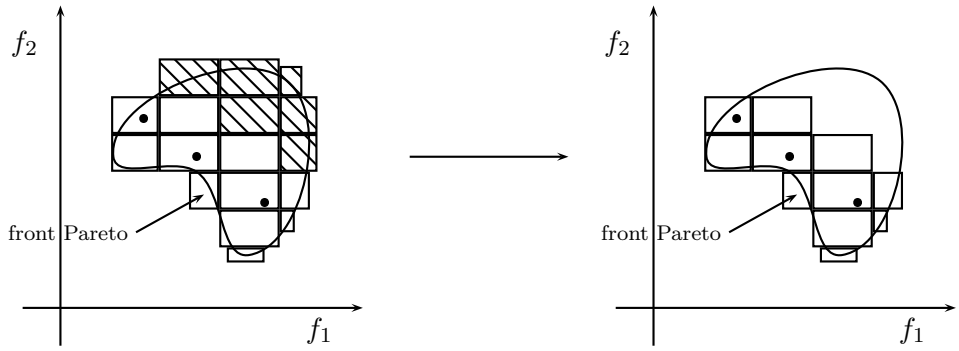


FIG. 7.1 – *Un exemple de sélection Pareto.*

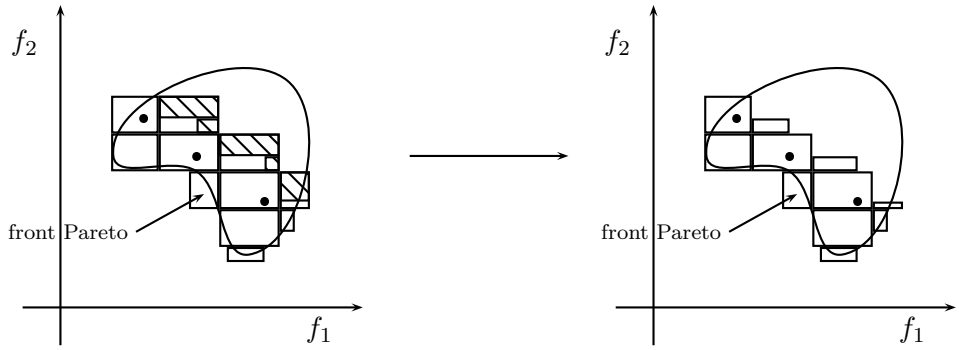


FIG. 7.2 – *Un exemple de sélection Pareto améliorée.*

résultat de cette réduction n'est plus une boîte, mais une union de boîtes. Le processus de sélection a pour but essentiel de diminuer la taille de la population en ne sélectionnant que les individus les mieux adaptés, il n'est donc pas conseillé de créer de nouveaux individus pendant cette phase.

7.3 Méthodes d'instanciation des individus

Nous avons vu dans la section précédente qu'une fois que la population a atteint sa taille maximale, nous appliquons une procédure de sélection Pareto dans le but d'éliminer les individus dominés grâce à la relation de PS-dominance présentée à la Section 6.3.3. La PS-dominance opère sur un point et un ensemble, or nous n'avons pas encore de point à ce moment de la recherche. Pour créer ces points, nous déclenchons un processus d'instanciation qui essaiera de trouver un point réalisable $\overrightarrow{x}$ contenu dans un individu $\overrightarrow{[x]}$ en déterminant pour chaque variable une valeur contenue dans son intervalle de définition $[x]$.

Remarquons que trouver un tel point réalisable correspond à résoudre un *problème de satisfaction de contraintes* [Tsang, 1993], tâche réputée très difficile dans le cas général.

Dans *PICPA* (cf. Chapitre 6), nous employons un opérateur d'instanciation étendant une instanciation partielle à une instanciation complète. Malheureusement, l'efficacité d'un tel opérateur dépend fortement du nombre de variables du problème. Dans cette section, nous présentons deux algorithmes d'instanciation basés sur des techniques de recherche locale. Le premier, appelé *Recherche Locale Continue*, ressemble beaucoup à la Stratégie d'évolution (1+1) (ES) [Back *et al.*, 1991]. Le second, appelé *Recherche Locale Dichotomique*, utilise un procédé de découpes dichotomiques pour guider un opérateur de recherche local. Cette section présente plus en détail ces deux algorithmes d'instanciation et donne une comparaison des deux algorithmes.

7.3.1 L'opérateur de recherche locale continue (CLS)

Rappelons qu'un individu $\overrightarrow{[x]}$ de longueur n est défini à partir de n intervalles localement consistants. Donnons-nous un tel individu, un opérateur de recherche locale continue (CLS) cherche un point réalisable $\overrightarrow{x} \in \overrightarrow{[x]}$.

Le concept de CLS ressemble beaucoup à la stratégie d'évolution (1+1) présentée dans [Back *et al.*, 1991]. Démarrant avec un point $\overrightarrow{x^0}$ non réalisable, CLS génère de manière itérative une série de points voisins de meilleure qualité (violant moins de contraintes) $\overrightarrow{x^0}, \overrightarrow{x^1}, \cdots, \overrightarrow{x^i}, \overrightarrow{x^{i+1}}, \cdots$ Le point initial peut être généré en assignant à chaque variable une valeur prise aléatoirement dans son intervalle de définition.

Plus concrètement, l'algorithme CLS est défini par trois éléments centraux : 1) une *fonction d'évaluation* pour mesurer la qualité des points, 2) une *fonction de voisinage* pour générer les points voisins et 3) une *stratégie de mouvement* décidant de l'acceptation ou du rejet d'un point voisin.

Fonction d'évaluation : Soit C un ensemble de contraintes et $\overrightarrow{x}$ un point de $\mathbb{R}^n$, la fonction d'évaluation est simplement définie comme suit :

$$eval(\overrightarrow{x}, C) = \sum \{deg(c) \mid c \in C,\ c \text{ n'est pas satisfait par } \overrightarrow{x}\}$$

où $deg(c)$ est la fonction évaluant le niveau de violation de la contrainte c. Ainsi, cette fonction calcule la distance entre l'évaluation de la contrainte violée et son domaine de faisabilité. Plus la valeur de l'évaluation de la contrainte est éloignée du domaine de faisabilité et plus le niveau de violation sera élevé. Évidemment, la valeur minimale de la fonction $eval(\overrightarrow{x}, C)$ est 0, correspondant à un point satisfaisant toutes les contraintes de C. Dans la suite de cette thèse, nous utiliserons $eval(\overrightarrow{x})$ en remplacement de $eval(\overrightarrow{x}, C)$.

Fonction de voisinage : Soit un point $\overrightarrow{x}$, un point voisin $\overrightarrow{x'}$ est construit en ajoutant des nombres aléatoires à $\overrightarrow{x}$. Dans la suite, x_i désigne la $i^{ème}$ composante du vecteur $\overrightarrow{x}$. La fonction de voisinage est définie formellement de la manière suivante :

$$\begin{aligned} x'_i &= x_i + (x_i^{(U)} - x_i^{(L)}) \times \mathbf{N}_0(\delta, \sigma_i) \\ \sigma'_i &= s(x_i, \sigma_i) \end{aligned}$$

où $x_i^{(U)}$ et $x_i^{(L)}$ correspondent respectivement à la borne supérieure et inférieure de la $i^{ème}$ composante, $\mathbf{N}_0$ désigne un vecteur de nombres aléatoires indépendants de moyenne 0, δ la loi de probabilité, σ_i $i \in \{1, \cdots, n\}$ l'écart type (ou facteur de perturbation), et s la fonction d'évolution de $\sigma \in \mathbb{R}^n$ au cours du temps.

Pour obtenir une fonction de voisinage complète, nous devons implémenter trois composants : la loi δ, le vecteur σ, et la fonction $s()$.

Le vecteur $\mathbf{N}_0$ est calculé grâce aux fonctions δ et σ. De plus, comme $\mathbf{N}_0$ est un vecteur de nombres aléatoires indépendants, une valeur aléatoire, nécessaire à δ, est calculée au préalable. Chaque élément de $\mathbf{N}_0$ peut être calculé de la manière suivante :

$$\mathbf{N}_0(\delta, \sigma) = (\delta(r_0, \sigma_0), \cdots, \delta(r_n, \sigma_n)) \text{ où } r_i \text{ est une valeur aléatoire}$$

Il est clair que $\mathbf{N}_0$ dépend fortement de la loi de probabilité δ. Dans cette étude, nous utilisons plusieurs lois δ, et nous décrivons les deux meilleures retenues après nos expérimentations.

La première, qui est aussi la plus classique, est une fonction normale δ_G (*Laplace-Gauss*).

$$\delta_G(r_i, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \times \exp\left(-\frac{r_i^2}{2\sigma_i^2}\right)$$

La seconde utilise une fonction polynomiale δ_P.

$$\delta_P(r_i, \sigma_i) = \begin{cases} (2r_i)^{\frac{1}{\sigma_i+1}} - 1, & \text{if } r_i < 0.5 \\ 1 - [2(1-r_i)]^{\frac{1}{\sigma_i+1}}, & \text{if } r_i \geq 0.5 \end{cases}$$

Cette fonction, présentée dans [Deb and Goyal, 1996], a été utilisée pour implémenter des opérateurs de mutation dans des algorithmes génétiques, ceux-ci ont obtenu des résultats expérimentaux très intéressants.

Le vecteur de perturbation σ doit aussi être défini. Dans ces travaux, nous utilisons des valeurs initiales déjà testées par d'autres chercheurs (cf. tableau 7.1). La fonction d'évolution $s()$ utilisée par l'algorithme est aussi un élément important. Cette fonction fait évoluer σ au cours du temps. Sans elle, l'algorithme utilise une seule loi de probabilité fixée, donc le voisinage sera pauvre et peu diversifié.

L'algorithme CLS utilise des lois de probabilité pour ajouter des quantités aléatoires aux variables. Notre approche étant très similaire aux algorithmes (1+1) ES ([Back *et al.*, 1991]), il est clair que les atouts de ces algorithmes sont exploités dans notre méthode CLS.

Pour tous nos opérateurs de recherche locale, nous utilisons la fonction $s()$ décrite dans [Back *et al.*, 1991].

$$s(\sigma^t) \;=\; \sigma^{t+1} \;=\; \begin{cases} c_d \times \sigma^t & \text{, si } p_s^t < \frac{1}{5} \\ c_i \times \sigma^t & \text{, si } p_s^t > \frac{1}{5} \\ \sigma^t & \text{, si } p_s^t = \frac{1}{5} \end{cases}$$

De plus, nous utilisons les paramètres de Schwefel présentés dans [Back *et al.*, 1991] : $c_d = 0.82$ et $c_i = \frac{1}{0.82}$. Nous appelons par la suite fonction de *Back_Schwefel*, la fonction

$s()$ présentée précédemment, où les valeurs des paramètres c_d et c_i sont celles suggérées par Schwefel.

Comme de nombreux choix sont possibles pour ces fonctions, nous ne pouvons étudier toutes les combinaisons. Nous avons donc décidé de garder quelques alternatives représentatives dans le but d'expliquer nos choix futurs. Le tableau 7.1 récapitule les trois meilleures combinaisons (appelées CLS_G, CLS_E et CLS_P dans la suite) gardées pour nos expérimentations (cf. Section 7.3.3).

	CLS_G	CLS_E	CLS_P
Loi de probabilité : δ	Laplace-Gauss : δ_G	Laplace-Gauss : δ_G	Polynomiale : δ_P
Vecteur de perturbation : σ	$(\frac{1.22}{n}, \ldots, \frac{1.22}{n})$	$(\frac{1.22\times\|x_i\|}{n}, \ldots, \frac{1.22\times\|x_n\|}{n})$	$(20, \cdots, 20)$
Fonction d'évolution de σ : $s()$	Fonction de *Back_Schwefel*	Fonction de *Back_Schwefel*	Fonction de *Back_Schwefel*

TAB. 7.1 – *Opérateurs de recherche locale continue*

Stratégie de mouvement : Soient un point $\overrightarrow{x}$ et un point $\overrightarrow{x'}$ issu de son voisinage. Une stratégie de mouvement définit les règles utilisées pour décider ou non du mouvement du point courant vers le point voisin. Dans ce chapitre, nous utilisons une règle à la manière des méthodes de descente qui n'acceptent que les voisins améliorant le point courant. Plus formellement,

$$\begin{aligned} \overrightarrow{x^{i+1}} &= \overrightarrow{x'^i} \text{ si } eval(\overrightarrow{x'^i}) < eval(\overrightarrow{x^i}) \\ &= \overrightarrow{x^i} \text{ sinon} \end{aligned}$$

Les algorithmes CLS étant des processus itératifs stochastiques, il est nécessaire de définir un critère d'arrêt pour assurer la terminaison de l'algorithme. Nous pouvons, par exemple, stopper l'algorithme lorsque le premier point réalisable est trouvé, lorsqu'aucune amélioration n'est observée pendant un nombre fixe d'itérations, ou lorsqu'un nombre d'itérations fixé au préalable est atteint par l'algorithme.

7.3.2 Recherche locale dichotomique (DLS)

L'algorithme CLS vu précédemment est simple à implémenter et donne de très bons résultats dans beaucoup de cas. Cependant, ses performances peuvent être améliorées si CLS est guidé vers des zones plus *prometteuses* de l'espace de recherche. C'est pourquoi nous introduisons un schéma de recherche dichotomique permettant d'identifier ces zones prometteuses. Simplement, DLS (Recherche Locale Dichotomique) répète la séquence suivante d'actions : *Bissection - ICP - Choix - CLS*.

Soit $[\overrightarrow{x}]$ un individu de n variables avec n intervalles localement consistants, nous prenons une variable et coupons en deux son intervalle actuel. Cette opération nous conduit à l'obtention de deux nouveaux individus $[\overrightarrow{a}]$ et $[\overrightarrow{b}]$. Nous appliquons ensuite notre algorithme

de propagation de contraintes sur des intervalles ICP (cf. Section 6.2) aux deux nouveaux individus. Après l'application des algorithmes d'ICP, trois situations sont possibles :

1. Aucun des deux individus n'est consistant (les applications d'ICP ont conduit à l'obtention d'un intervalle vide). Dans ce cas, aucun point réalisable n'existe pour le problème initial.

2. Seulement un des deux individus est consistant. Dans ce cas, nous abandonnons l'individu inconsistant et appliquons CLS à l'individu consistant. Si CLS ne parvient pas à calculer un point réalisable, nous redémarrons la phase de bissection.

3. Les deux individus sont consistants. Dans ce cas, un des deux individus est choisi pour les phases de bissection à venir. Pour réaliser ce choix, nous appliquons CLS à chacun des nouveaux individus et gardons le meilleur d'entre eux, c'est-à-dire celui ayant un degré moindre de violation des contraintes. L'individu qui n'a pas été sélectionné est oublié, et nous continuons avec l'individu restant.

Nous pouvons itérer ce processus heuristique jusqu'à l'obtention d'un point réalisable. Cependant, lorsque les contraintes deviennent trop difficiles à satisfaire, le nombre de coupes nécessaires pour découvrir un point réalisable devient très important. Nous devons donc déterminer un critère d'arrêt pour limiter le processus de découpe. Nous rappelons que *PICPA-II* coupe principalement dans les intervalles liés aux domaines de définition des objectifs et projette ces découpes sur les domaines des variables grâce à la propagation de contraintes (cf Section 6.2). Ainsi, les domaines des variables sont souvent plus grands que ceux des objectifs. Mais, dans ce procédé *dichotomique*, nous coupons dans les domaines des variables. Un critère d'arrêt élaboré est alors : nous appliquons le procédé dichotomique tant que le plus grand domaine des variables est plus grand que le plus grand domaine des fonctions objectifs.

D'autres critères d'arrêt sont possibles. Par exemple, borner le nombre de découpes en fixant un nombre maximal de découpes autorisées. Ce type de critère d'arrêt nous paraît mal adapté, car il ne tient pas compte de la topologie du problème. Ainsi, nous nous sommes dirigés vers des critères d'arrêt utilisant cette topologie, en nous basant sur la taille des intervalles de l'individu considéré. Il existe beaucoup de façons de définir un critère d'arrêt en se basant sur la taille des intervalles : fixer une taille minimale, calculer la moyenne de la taille de chaque intervalle, ... Remarquons que chaque critère est différent en fonction de l'individu considéré car les intervalles ne sont pas les mêmes. Quand un critère d'arrêt est rencontré, nous pouvons appliquer un opérateur de recherche locale continue vu dans la section précédente.

Notre méthode de *recherche locale dichotomique* peut être considérée comme une heuristique d'exploration d'une seule branche de l'arbre de recherche pour un certain nombre de nœuds. Cette méthode s'arrête si un domaine vide ou réduit au singleton est découvert, ou si le critère d'arrêt défini est rencontré.

Bien sûr, les domaines originaux de l'individu initial restent inchangés. En effet, le but de l'algorithme DLS est juste de trouver un point réalisable, il ne doit pas altérer

Algorithme 12 Algorithme général de la *recherche locale dichotomique*

Soit I l'individu à instancier
$I_{inst} \longleftarrow I$
Répéter
Couper I_{inst} pour obtenir I_1 et I_2
$ISet \longleftarrow \{(j, CLS(j)) \mid j \in \{ICP(I_1), ICP(I_2)\}, j$ est localement consistant$\}$
$I_{min} \longleftarrow \min_{eval(j_{cls})}\{(j, eval(j_{cls})) \mid (j, j_{cls}) \in ISet\}$
$I_{inst} \longleftarrow j$ s.t $I_{min} = (j, eval(j_{cls})),\ (j, j_{cls}) \in ISet$
Jusqu'à (Critère_Arrêt) *ou* (Domaine_Vide_Trouvé) *ou* (Point_Réalisable_Trouvé)

l'individu initial. Le pseudo-code de l'algorithme 12 montre le fonctionnement de notre méthode de *recherche locale dichotomique* :

On désigne par *eval*() la fonction d'évaluation, présentée dans la Section 7.3.1, permettant de quantifier le degré de violation des contraintes.

7.3.3 Comparaison de CLS et DLS

Dans cette section, nous évaluons sur des problèmes test les différents algorithmes de recherche locale (CLS_G, CLS_E, CLS_P avec et sans DLS) présentés précédemment. Rappelons que, pour ces opérateurs, nous cherchons à trouver le premier point réalisable et non pas à déterminer toutes les solutions d'un problème multiobjectif.

Les problèmes test

Pour ces expérimentations, nous utilisons deux problèmes test. Le premier a été utilisé dans la chapitre 6, nous le rappelons ici :

$$\text{TNK} \begin{cases} \text{Soient} & c_1(x) \equiv x_1^2 + x_2^2 - 1 - 0.1\cos(16\arctan(\frac{x_1}{x_2})) \geq 0 \\ & c_2(x) \equiv (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5 \\ \text{et} & x_1, x_2 \in [0 \cdots \pi] \end{cases}$$

Notons que pour ce problème, seules les contraintes ont été conservées, les objectifs ont été supprimés car nous ne cherchons que le premier point réalisable. Le second problème est issu d'une application réelle du domaine de la robotique [Jaulin *et al.*, 2001].

$$\text{RBT} \begin{cases} \text{Soient} & \alpha_{j1} = (l_1 \times \cos(t_{j1}) - l_2 \times \sin(t_{j2} + t_{j3})) \times \cos(t_{j1}) \\ & \alpha_{j2} = (l_1 \times \cos(t_{j1}) - l_2 \times \sin(t_{j2} + t_{j3})) \times \sin(t_{j1}) \\ & \alpha_{j3} = l_1 \times \sin(t_{j1}) + l_2 \times \cos(t_{j2} + t_{j3}) \\ \text{et} & l_1 \in [1.4 \cdots 1.42], \quad l_2 \in [0.98 \cdots 1] \\ & t_{j1} \in [-\pi \cdots \pi], \quad t_{j2}, t_{j3} \in [\frac{-\pi}{4} \cdots \frac{\pi}{4}] \end{cases}$$

La difficulté de ce problème est adaptable. En effet, elle croît lorsque l'on augmente le nombre de variables et de contraintes. Pour nos expérimentations, nous utilisons trois instances distinctes de RBT : la première instance comprend 3 variables et 3 contraintes

(RBT_1), la seconde 6 variables et 6 contraintes (RBT_2) et la dernière 18 variables et 18 contraintes (RBT_3). Comme nous le verrons dans la prochaine section, les contraintes d'égalité utilisées dans ces problèmes sont très difficiles à résoudre pour un algorithme de recherche locale, car le nombre de points réalisables présents dans l'espace de recherche est très faible.

Résultats expérimentaux

Le tableau 7.2 récapitule les résultats obtenus pour les opérateurs de recherche locale continue présentés au tableau 7.1.

Pour ces expérimentations, nous comptons le nombre d'itérations nécessaires à chaque algorithme pour calculer un point réalisable (satisfaisant toutes les contraintes du problème). Le tableau 7.2 donne les moyennes de ces valeurs calculées sur 10 exécutions indépendantes[2]. Les symboles ?? signifient que l'algorithme a été stoppé manuellement au bout de 10 minutes. Dans ce cas, nous considérons que l'algorithme a échoué.

	CLS_G		CLS_E		CLS_P	
	Itérations	Echecs	Itérations	Echecs	Itérations	Echecs
TNK	8.4	0	6.4	0	7.3	0
RBT_1	198.7	0	142	0	164.7	0
RBT_2	1247.1	0	541.5	0	404.1	0
RBT_3	??	10	??	10	??	10

TAB. 7.2 – *Résultats des opérateurs de recherche locale continue.*

Comme nous le constatons dans le tableau 7.2, les trois opérateurs permettent d'obtenir des résultats comparables sur les problèmes faciles (TNK et RBT_1). Pour l'instance RBT_2, plus difficile à résoudre, CLP_P est trois fois plus rapide que CLS_G. Pour RBT_3 qui est aussi l'instance la plus difficile, aucune des trois méthodes n'arrive à trouver de solution réalisable.

Le tableau 7.3 récapitule les résultats obtenus par les opérateurs de recherche locale continue combinés avec notre processus de *découpes dichotomiques*. Nous obtenons donc plusieurs algorithmes de *recherche locale dichotomique (DLS)*.

Pour ces expérimentations, nous calculons deux valeurs : le nombre total d'itérations effectuées par la partie recherche locale, et le nombre de coupes faites par la partie dichotomique. Le tableau 7.3 donne les moyennes de ces valeurs calculées sur 10 exécutions indépendantes[2], chaque appel de recherche locale étant limité à 1000 itérations. Les résultats obtenus sont comparables jusqu'à ce que l'on considère l'instance RBT_3. Sur cette instance, l'algorithme $DLS + CLS_P$ obtient de meilleurs résultats que les deux autres algorithmes DLS.

Il est très intéressant de noter que, même s'il est difficile de comparer les résultats obtenus par DLS et un algorithme CLS pur, DLS combiné avec un opérateur CLS trouve

2. Nous effectuons 10 exécutions indépendantes à cause du caractère stochastique des algorithmes.

	$DLS + CLS_G$			$DLS + CLS_E$			$DLS + CLS_P$		
	Itérations	Coupes	Echecs	Itérations	Coupes	Echecs	Itérations	Coupes	Echecs
TNK	12.8	0	0	6.9	0	0	7.6	0	0
RBT_1	1	10	0	1	10	0	1	10	0
RBT_2	31	19	0	57.5	19	0	41.9	19	0
RBT_3	23818.1	41.4	2	21162.9	35.9	1	19258.2	32.4	1

TAB. 7.3 – *Résultats des opérateurs DLS+ recherche locale continue.*

une solution sur tous les problèmes traités ici. De plus, lorsqu'une coupe est effectuée, le nombre d'itérations de recherche locale nécessaires est réduit de façon radicale. Finalement, nous observons que DLS est plus robuste car il échoue plus rarement pour découvrir une solution sur ces instances.

Les algorithmes DLS développés dans cette section sont utilisés pour trouver un point réalisable. Bien que nous restreignions l'utilisation des algorithmes DLS à la résolution de problèmes CSP (trouver une solution), ils peuvent aussi être utilisés pour résoudre des problèmes d'optimisation continue sous contraintes.

Dans la suite de cette thèse, nous choisissons l'algorithme $DLS + CLS_P$ comme opérateur d'instanciation nécessaire pour l'application de la PS-dominance dans le processus de sélection. En effet, c'est cette version qui a procuré les meilleurs résultats sur les problèmes testés ici. Notons que le paramètre *d'effort de recherche* présenté à la Section 7.4 correspond ici au nombre d'itérations de recherche locale utilisé pour limité l'algorithme DLS (cf. expérimentations précédentes).

7.4 Un algorithme combinant population et propagation de contraintes de façon améliorée : *PICPA-II*

PICPA-II combine un processus d'ICP (cf. Section 6.2) avec une sélection Pareto de boîtes améliorée (cf. Section 7.2) dans un seul algorithme. *PICPA-II* utilise ainsi les mêmes concepts que ceux introduits dans *PICPA*.

Une fois que la population a atteint sa taille maximale, un processus d'instanciation est déclenché dans le but de trouver des instanciations complètes et réalisables pour chaque individu $[\overrightarrow{x}]$ de la population. Plus précisément, une valeur particulière satisfaisant les contraintes tout en restant incluse dans la boîte sera recherchée, et ceci pour tous les individus de la population. Ce processus d'instanciation utilise une méthode de recherche locale, son fonctionnement a fait l'objet de la section 7.3.

Comme cette opération requiert un temps de calcul très important, *PICPA-II* utilise le paramètre *d'effort de recherche* (correspondant dans *PICPA-II* à un nombre d'itérations de recherche locale) comme un critère d'arrêt permettant de stopper le processus d'instanciation si aucune solution réalisable n'a été trouvée pendant un certain laps de temps. Ainsi, après cette étape, certains individus sont complètement instanciés, et contiennent

une solution réalisable $\overrightarrow{x}$, tandis que d'autres restent non instanciés. Les solutions issues de cette phase d'instanciation sont enregistrées dans une structure de données annexe, si bien que les individus de la population courante ne sont pas modifiés.

Nous donnons maintenant le pseudo-code de *PICPA-II* dans l'algorithme 13.

Algorithme 13 Pseudo-code de l'algorithme *PICPA-II*.

```
Initialiser la population avec un unique individu localement consistant
Tant que 0 < |Population| < Max Population Size faire
    Tant que 0 < |Population| < Max Population Size faire
        - Sélectionner un individu (parent) et réaliser une bissection selon
          un des objectifs, conduisant à deux individus (enfants) distincts
        - Réduction des domaines des enfants (ICP)
        - Mise à jour de la population :
                (a) Retirer le père
                (b) Ajouter les enfants localement consistants
    FinTantQue
    - Instanciation potentielle de chaque individu (algorithme de recherche locale)
    - Processus de sélection Pareto améliorée (PS-dominance)
FinTantQue
```

Nous observons que l'algorithme *PICPA-II*, comme l'algorithme *PICPA*, nécessite seulement deux paramètres : la taille de la population et l'effort de recherche.

PICPA-II est une extension de l'algorithme *PICPA*. Il existe cependant une différence fondamentale entre *PICPA* et *PICPA-II* : *le déterminisme*. *PICPA* est effectivement un algorithme complètement déterministe alors que *PICPA-II*, qui intègre un nouvel opérateur d'instanciation à base de recherche locale, ne l'est plus.

7.5 Résultats expérimentaux

Dans cette section, nous effectuons des expérimentations sur *PICPA-II* (utilisant $DLS+CLS_P$ comme procédure d'instanciation) sur des problèmes test de la littérature. Comme pour les expérimentations effectuées avec *PICPA* (cf. Section 6.4), nous utilisons *NSGA-IIc* comme algorithme de référence pour évaluer les résultats de notre approche. Certains des problèmes que nous allons utiliser ont déjà été le sujet d'expérimentations à la Section 6.4. Nous pourrons donc comparer les résultats de *PICPA-II* avec ceux obtenus par *PICPA*. Pour ces expérimentations, nous utilisons les paramètres suivants :

– Pour *NSGA-IIc*, nous utilisons les valeurs des paramètres données [Deb *et al.*, 2001]. Plus précisément, nous fixons pour le croisement binaire simulé [Deb and Agrawal, 1995], $n_c = 20$, et pour l'opérateur de mutation polynomiale, $n_m = 20$. La probabilité de croisement est fixée à 0.9 et la probabilité de mutation à 0.15. La taille de la population, ainsi que le nombre maximal de générations, sont choisis en fonction de la difficulté du problème.

– Pour *PICPA-II*, nous fixons la taille de la population à 300 et l'effort de recherche à 200.

Comme *PICPA-II* utilise plusieurs opérateurs différents (propagation de contraintes, recherche locale, concepts évolutionnaires), il n'est pas possible de fixer des valeurs identiques aux paramètres des deux algorithmes en compétition. En effet, celles-ci fausseraient les résultats. Nous avons donc choisi des valeurs permettant aux algorithmes de résoudre le problème avec des temps de calcul du même ordre.

PICPA-II et *NSGA-IIc* étant des algorithmes stochastiques, nous les exécutons dix fois sur chaque problème test, et gardons la meilleure exécution pour nos comparaisons. Nous observons que les résultats de *PICPA-II* présentent de faibles écart-types par rapport aux résultats de *NSGA-IIc*.

7.5.1 Le problème de Binh et Korn

Le premier problème utilisé a été introduit par Binh et Korn dans [Binh and Korn, 1997] :

$$\text{BNH} \begin{cases} \text{Minimiser} & f_1(x) = 4x_1^2 + 4x_2^2 \\ \text{Minimiser} & f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 \\ \text{t.q.} & c_1(x) \equiv (x_1 - 5)^2 + x_2^2 \leq 25 \\ & c_2(x) \equiv (x_1 - 8)^2 + (x_2 + 3)^2 \geq 7.7 \\ \text{et} & x_1 \in [0..5] \\ & x_2 \in [0..3] \end{cases}$$

Pour les expérimentations sur BNH, nous utilisons pour *NSGA-IIc* une population de taille 150 et un nombre maximal de générations fixé à 500. Le front Pareto optimal ainsi que les bornes calculées par *PICPA-II* sont présentés à la figure 7.3(a). La figure 7.3(b) montre les surfaces de compromis calculées par *PICPA-II* et *NSGA-IIc*.

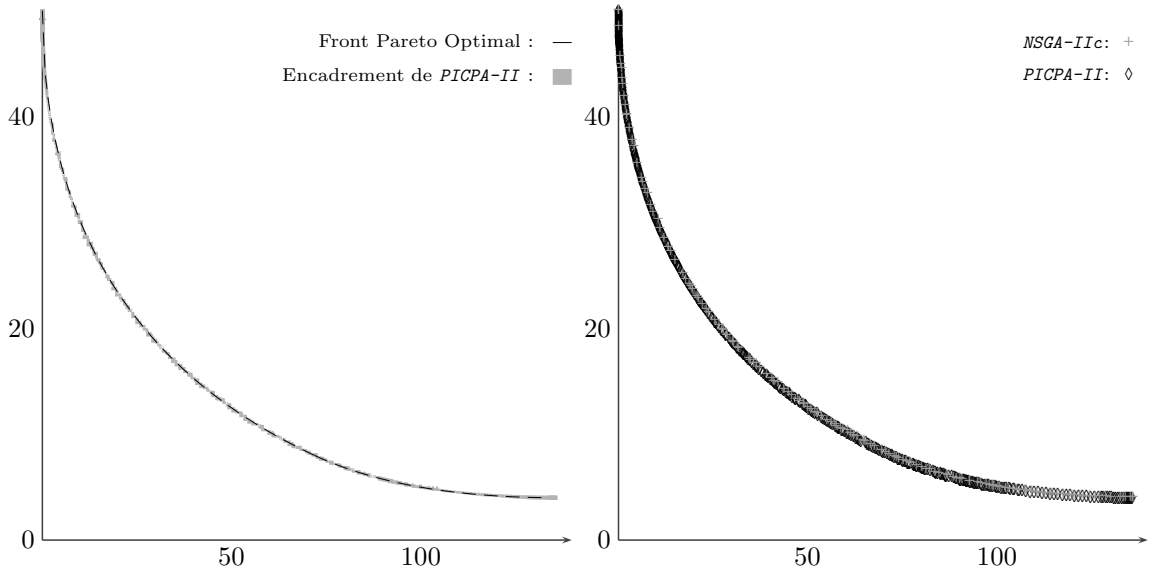
D'après ces figures, nous observons que 1^o) les bornes de *PICPA-II* encadrent presque parfaitement le front Pareto optimal et que 2^o) *PICPA-II* et *NSGA-IIc* trouvent des solutions approchées de qualité comparable.

7.5.2 Le problème test de Osyczka et Kundu

Pour notre seconde expérimentation, nous reprenons le problème de Osyczka et Kundu déjà utilisé à la Section 6.4.

Comme pour les expérimentations sur BNH (cf. Section 7.5.1), nous fixons pour *NSGA-IIc* la taille de la population à 150 et le nombre de générations à 500.

D'après la figure 7.4(a), nous observons que les bornes de *PICPA-II* sont globalement très proches du front Pareto optimal, excepté pour quelques zones. Ces zones moins bien cernées sont dues au pourcentage élevé de contraintes additives. À la figure 7.4(b), nous observons que la qualité des deux ensembles de solutions non dominées découverts par *PICPA-II* et *NSGA-IIc* est très proche. Remarquons, que sur cette instance de problème, des exécutions différentes de *NSGA-IIc* donnent des résultats de qualité très variable.



(a) Bornes du front Pareto optimal.

(b) Ensemble de solutions non dominées.

FIG. 7.3 – *Résultats expérimentaux pour le problème BNH.*

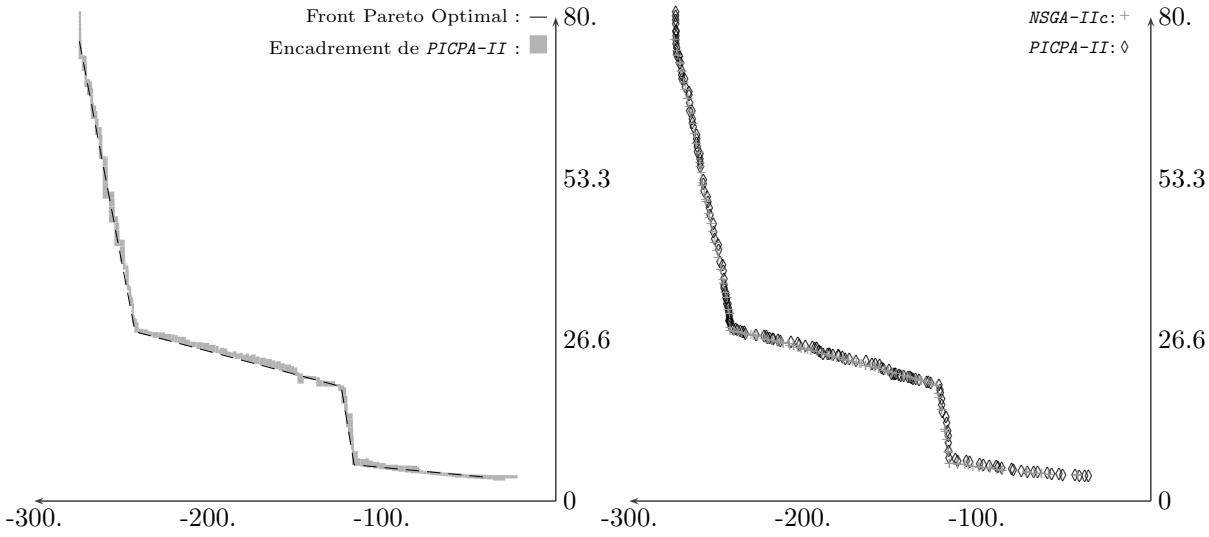
En comparant les bornes obtenues par ***PICPA-II*** avec celles calculées par ***PICPA*** à la figure 6.6(a), nous observons que ***PICPA-II*** améliore la précision de l'encadrement. De même, les solutions trouvées par ***PICPA*** à la figure 6.6(b) sont de moins bonne qualité que celles calculées par ***PICPA-II***. Cette différence est très nette sur la partie horizontale du front Pareto.

7.5.3 Un problème test contraint (CTP)

Pour ces expérimentations, nous utilisons un des générateurs de problèmes test présentés dans [Deb *et al.*, 2001] et [Deb, 2001]. Comme ces problèmes ont une difficulté réglable, nous utilisons une fonction $g()$ basée sur la fonction de Rastrigin :

$$\text{CTP}\begin{cases} \text{Minimiser} & f_1(x) = x_1 \\ \text{Minimiser} & f_2(x) = g(x) - f_1(x) \\ \text{t.q.} & g(x) = 91 + \sum_{i=2}^{10}\left(x_i^2 - 10\cos(4\pi x_i)\right) \\ & C_1(x) \equiv \cos(\theta)(f_2(x) - e) - \sin(\theta) f_1(x) \geq \\ & \qquad a|\sin(b\pi(\sin(\theta)(f_2(x) - e) + \cos(\theta) f_1(x))^c)|^d \\ \text{et} & x_1 \in [0..1] \\ & x_{i,i>1}, \in [-5..5] \end{cases}$$

Pour ces expérimentations, nous utilisons le dernier générateur CTP décrit dans [Deb,



(a) Bornes du front Pareto optimal.

(b) Ensemble de solutions non dominées.

FIG. 7.4 – *Résultats expérimentaux pour le problème OSY.*

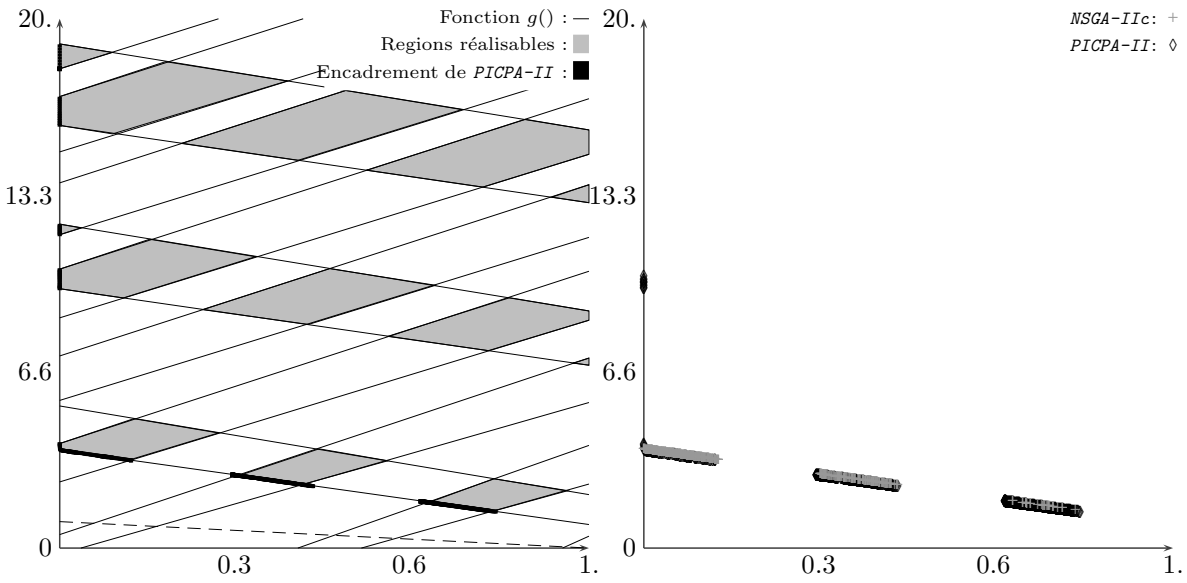
2001] (c.à.d CTP8) qui construit des instances très difficiles à résoudre. Ce problème a déjà été présenté à la section 6.4, mais ici, nous utilisons une fonction de Rastrigin à 10 variables, ce qui augmente la difficulté du problème. CTP8 est composé de plusieurs régions réalisables non connexes. Dans CTP8, nous avons deux contraintes à satisfaire, ce qui n'est pas le cas dans les autres problèmes de type CTP. Nous donnons ci-après les paramètres utilisés pour générer les contraintes de notre instance CTP8 :

$$C_1 \ : \ \theta = 0.1\pi, \quad a = 40, \quad b = 0.5, \quad c = 1, \quad d = 2, \quad e = -2$$

$$C_2 \ : \ \theta = -0.05\pi, \quad a = 40, \quad b = 2, \quad c = 1, \quad d = 6, \quad e = 0$$

Pour ces expérimentations, nous fixons pour *NSGA-IIc*, la taille de la population à 300 et le nombre de générations à 1000. Les paramètres de *PICPA-II* restent les mêmes que dans les sections précédentes.

La figure 7.5(a) montre les zones réalisables de l'espace des objectifs, les zones non connexes du front Pareto optimal ainsi que les bornes calculées par *PICPA-II*. La figure 7.5(b) montre les ensembles de solutions découverts par *NSGA-IIc* et *PICPA-II*. D'après la figure 7.5(a), nous constatons là aussi que les bornes de *PICPA-II* sont très précises. Pour ce problème, qui est aussi le plus difficile testé ici, *NSGA-IIc* et *PICPA-II* trouvent des solutions sur toutes les zones non connexes du front Pareto optimal. La figure 7.5(b) montre que les qualités des deux ensembles de solutions découverts par *PICPA-II* et *NSGA-IIc* sont très proches. Remarquons toutefois que, sur ce problème, des exécutions différentes de *NSGA-IIc* donnent des résultats de qualité très variable. Sur certaines exécutions,



(a) Bornes du front Pareto optimal.

(b) Ensemble de solutions non dominées.

FIG. 7.5 – *Résultats expérimentaux pour le problème CTP8.*

NSGA-IIc ne découvre aucune solution sur une ou plusieurs composantes connexes du front Pareto optimal.

Nous avons utilisé ici une fonction de Rastrigin comportant 10 variables, alors que nous n'en utilisions que 5 pour les expérimentations avec *PICPA*. Bien que le problème soit plus difficile à résoudre, nous observons que les bornes calculées par *PICPA-II* sont plus précises que celles obtenues par *PICPA* (cf. figure 6.8(a)). Comme les solutions calculées par les deux algorithmes sont de qualité équivalente, nous pouvons donc conclure que *PICPA-II* améliore les résultats de *PICPA* sur le problème CTP8.

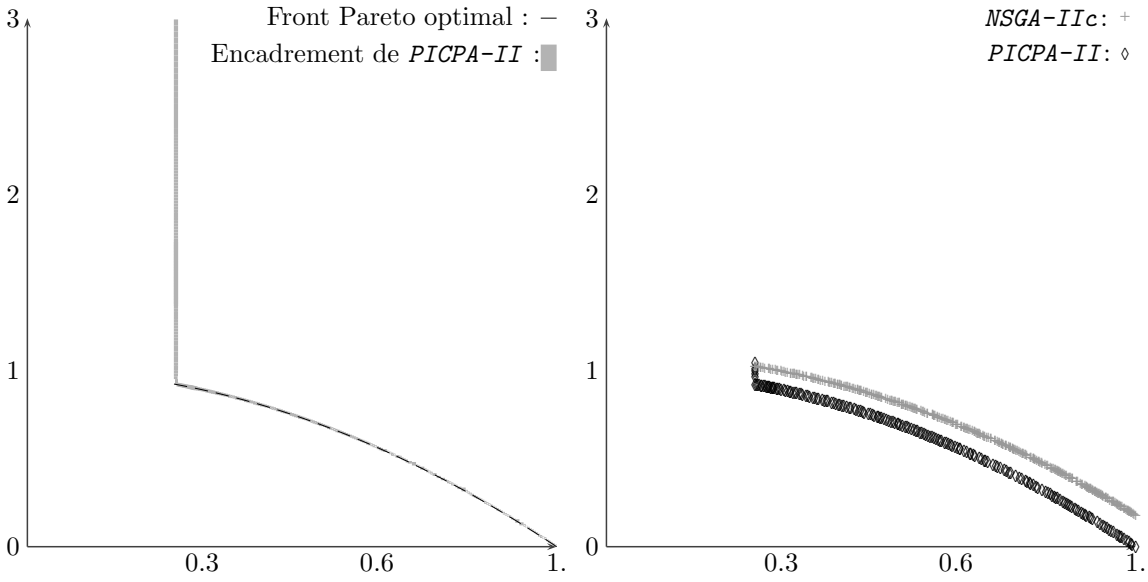
7.5.4 Un problème introduit par Zitzler, Deb et Thiele

Pour notre dernière expérimentation, nous choisissons un problème test de dix variables sans contrainte. Ce problème fut présenté par Zitzler, Deb et Thiele [Zitzler *et al.*, 2000] :

$$\text{ZDT6} \begin{cases} & g(x) = 1 + 9\left(\frac{\sum_{i=2}^{10} x_i}{9}\right)^{0.25} \\ \text{Minimiser} & f_1(x) = 1 - \exp(-4x_1) \times \sin^6(6\pi \times x_1) \\ \text{Minimiser} & f_2(x) = g(x) \times \left[1 - \left(\frac{f1(x)}{g(x)}\right)^2\right] \\ \text{et} & \forall i,\ x_i \in [0 \cdots 1] \end{cases}$$

Comme pour les deux premières expérimentations (cf. sections précédentes), nous fixons, pour *NSGA-IIc*, la taille de la population à 150 et le nombre de générations à

500.



(a) Bornes du front Pareto optimal.

(b) Ensemble de solutions non dominées.

FIG. 7.6 – *Résultats expérimentaux pour le problème ZDT6.*

Sur la figure 7.6(a), nous observons que les bornes de *PICPA-II* encadrent de manière très fine le front Pareto optimal. Bien qu'il n'y ait aucune contrainte, les opérateurs de projection utilisés par *PICPA-II* restent efficaces sur ce problème test. La figure 7.6(b) montre que les points découverts par *PICPA-II* dominent ceux trouvés par *NSGA-IIc*. Les résultats de *NSGA-IIc* calculés ici sont semblables à ceux exposés dans [Deb and Goel, 2001] par les auteurs de l'algorithme. Nous observons que les solutions de *NSGA-IIc* sont éloignées du front Pareto optimal alors que celles de *PICPA-II* en sont très proches.

Notons que, sur ce problème, les algorithmes *SPEA* [Zitzler and Thiele, 1999] et *PICPA-II* trouvent des résultats similaires, c.à.d que les deux algorithmes obtiennent de meilleurs résultats que *NSGA-IIc* sur cette instance de problème.

7.6 Conclusion

Dans ce chapitre, nous avons présenté *PICPA-II*, une amélioration de *PICPA* (cf. Chapitre 6) pour traiter les problèmes multiobjectifs continus avec contraintes. *PICPA-II* combine, à l'instar de *PICPA*, des méthodes de propagation de contraintes sur intervalles avec des concepts évolutionnaires (population et sélection) et de la recherche locale. L'innovation dans *PICPA-II* réside dans le fonctionnement de ses opérateurs de sélection et d'instanciation. Ce dernier allie recherche locale et dichotomique en une seule procédure : *recherche locale dichotomique (DLS)*. Pourvu de cet opérateur, *PICPA-II* améliore tous

les résultats obtenus par *PICPA* sur tous les types de problèmes, avec ou sans contraintes, avec des contraintes linéaires ou non linéaires. En contrepartie, l'algorithme n'est plus déterministe. Le non déterminisme de l'algorithme n'influe en aucun cas sur la garantie des résultats fournis. Il implique juste que, sur plusieurs exécutions, les bornes calculées par *PICPA-II* seront un peu différentes.

PICPA-II conserve la propriété essentielle de *PICPA* et permet de borner le front Pareto optimal tout en calculant des solutions approchées. Les expérimentations effectuées sur des instances de problèmes issus de la littérature ont montré l'efficacité de *PICPA-II* pour calculer une très bonne surface de compromis. Les temps de calcul n'ont été évoqués que très rapidement dans ce chapitre car ils ne constituent pas le centre de notre étude. Remarquons que, pour toutes les expérimentations effectuées, les paramètres sont fixés de manière à ce que les différents algorithmes en compétition s'exécutent dans des temps très proches (la différence pouvant aller de une seconde à une dizaine de secondes). La comparaison des temps de calcul nous paraît difficile à effectuer de manière équitable car les algorithmes en compétition ne donnent pas le même type de résultat. En effet, *PICPA-II* calcule non seulement des solutions mais aussi des bornes de ces solutions, à la différence des autres algorithmes en compétition, qui donnent seulement des solutions heuristiques.

PICPA-II améliore tous les résultats obtenus par *PICPA*. Nous pensons donc que ce schéma d'hybridation est prometteur et que *PICPA-II* peut encore être amélioré. En particulier, toutes les heuristiques (choix de l'objectif à couper, critère d'arrêt de la recherche locale, composants des algorithmes CLS) sont inspirées de méthodes classiques de la littérature. Il est sans doute possible d'adapter dans *PICPA-II* différentes stratégies utilisées pour la résolution de problèmes sous contraintes.

Nous pensons aussi que l'algorithme DLS utilisé dans ce chapitre comme opérateur d'instanciation pour *PICPA-II* peut être appliqué à des problèmes d'optimisation globale avec contraintes. Cette méthode souple intègre plusieurs opérateurs déterministes (bissection, propagation de contraintes) permettant de mieux guider la recherche. La performance de l'algorithme DLS pour l'optimisation globale sous contraintes fera l'objet d'une étude future.

Quatrième partie

Extensions et ouvertures

Chapitre 8

Frontière Pareto

Dans ce chapitre, nous présentons de nouvelles définitions établissant le lien entre les problèmes multiobjectifs et les enveloppes axe convexes d'ensembles. Nous énonçons et démontrons des propriétés permettant d'effectuer cette jonction, ouvrant ainsi de nouvelles perspectives de recherche. Nous traitons un exemple d'application réelle issu du domaine de l'automatique et présentons ainsi une nouvelle application de notre algorithme *PICPA-II*.

Sommaire

8.1 Introduction

Trouver le front Pareto optimal d'un problème multiobjectif est déjà une finalité en soi. Dans cette thèse, nous avons évoqué plusieurs algorithmes de résolution et proposé un autre type d'approche pour les traiter. Le but que nous nous fixions était de résoudre une instance de problème le plus efficacement possible.

Cependant, il existe d'autres manières d'utiliser les problèmes multiobjectifs. En effet, ceux-ci peuvent être utilisés pour modéliser des parties de problèmes plus généraux. Un problème initial pourrait par exemple se modéliser en fonction de plusieurs problèmes multiobjectifs. Ainsi, résoudre un problème multiobjectif ne suffirait pas pour résoudre le problème initial, mais cela y contribuerait grandement. Toutes les connaissances développées dans le cadre de l'optimisation multiobjectif peuvent donc être réutilisées dans ce nouveau domaine d'applications.

Dans ce chapitre, nous présentons une nouvelle utilisation des problèmes multiobjectifs et de leurs algorithmes de résolution. Notre but ici, n'est plus de résoudre un problème d'optimisation multiobjectif, mais de calculer une approximation de la frontière d'un ensemble fermé. L'approximation d'ensemble peut sembler éloignée du thème de l'optimisation multiobjectif et pourtant nous montrons dans ce chapitre que le lien existe. Pour approcher la frontière d'un ensemble, nous allons montrer qu'en résolvant plusieurs problèmes d'optimisation multiobjectifs, nous pouvons déterminer des parties de l'enveloppe axe-convexe de l'ensemble. Ainsi, grâce à cette enveloppe axe-convexe, nous obtenons une bonne approximation de l'ensemble. Nous rappelons d'abord quelques notions nécessaires à la présentation de nouvelles définitions. Nous présentons ensuite un exemple de ce nouveau type d'utilisation et donnons les résultats expérimentaux obtenus par `PICPA-II` sur deux instances d'un problème pratique entrant dans notre nouveau cadre [Dao *et al.*, 2003].

8.2 Définitions et propriétés

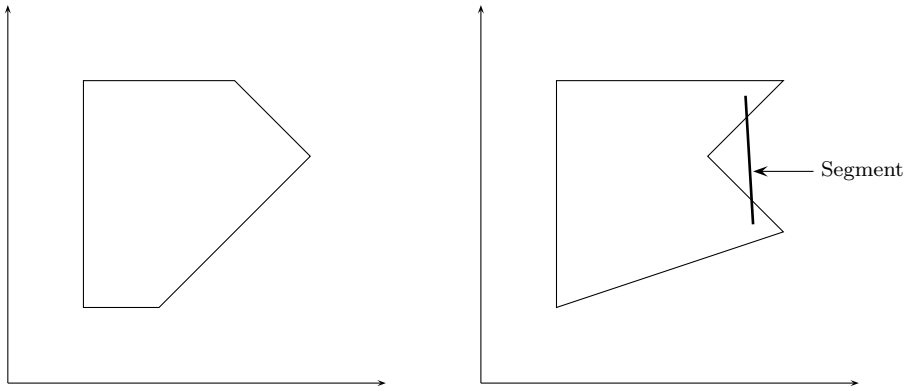
Nous allons maintenant donner les définitions et propriétés permettant de modéliser certains types de problèmes à l'aide de problèmes multiobjectifs.

8.2.1 La convexité

La convexité est une propriété assez forte caractérisant un ensemble. La définition de la convexité est donnée à la définition 15 page 14.

La figure 8.1 montre deux exemples de cas de convexité : l'ensemble de gauche est convexe, alors que celui de droite ne l'est pas.

Cette propriété sert de base à d'autres propriétés plus faibles mais permettant de meilleures approximations. Pour déterminer une approximation d'un ensemble, il est souvent plus facile de calculer le plus petit ensemble convexe contenant l'ensemble. Il est donc nécessaire de définir l'enveloppe convexe :



(a) Ensemble convexe. (b) Ensemble non convexe.

FIG. 8.1 – *Propriété de convexité.*

Définition 21 Enveloppe convexe : *Soient E un ensemble, et $A \subset E$. On appelle enveloppe convexe de A et l'on note conv(A) l'intersection de toutes les parties convexes de E contenant A. C'est, au sens de l'inclusion, le plus petit ensemble convexe contenant A.*

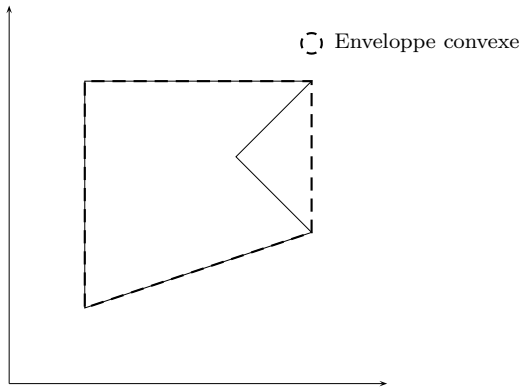


FIG. 8.2 – *Enveloppe convexe.*

La figure 8.2 montre l'enveloppe convexe de l'ensemble non convexe représenté à la figure 8.1(b).

8.2.2 La convexité de ligne et de colonne

La convexité de ligne ou de colonne est un cas particulier de convexité ne considérant que les segments parallèles à l'axe des abscisses (resp. des ordonnées) pour la convexité de ligne (resp. de colonne). Chacune de ces relations ne considère que les lignes et les colonnes du plan, elles ne sont utilisées que dans des espaces de dimension deux.

Définition 22 Convexité de ligne (de colonne) : *Un ensemble A est convexe en ligne, si et seulement si l'implication suivante est vérifiée :*

$$\forall x, y \in A^2, \qquad ord(y) = ord(x) \Rightarrow Segment(x, y) \subset A$$

De manière analogue, un ensemble A est convexe en colonne, si et seulement si :

$$\forall x, y \in A^2, \qquad abs(y) = abs(x) \Rightarrow Segment(x, y) \subset A$$

où $abs(x)$ est la fonction calculant l'abscisse de x, $ord(x)$ la fonction calculant l'ordonnée de x et $Segment(x, y)$ la fonction construisant le segment entre les points x et y. Il est clair que tout ensemble convexe est aussi convexe en ligne et en colonne.

8.2.3 L'axe-convexité

L'axe-convexité est une propriété plus faible que la convexité. Elle consiste à ne tester la convexité que sur des plans parallèles aux axes du repère. Elle généralise les notions de convexité de ligne et de colonne vues précédemment car elle fait intervenir tous les axes du repère.

Définition 23 Axe-convexité : *Soit A un ensemble connexe. A est axe-convexe si et seulement si l'égalité suivante est vérifiée :*

$$\forall \overrightarrow{[x]}, \qquad Sommets(\overrightarrow{[x]}) \subset A \Longleftrightarrow \overrightarrow{[x]} \subset A$$

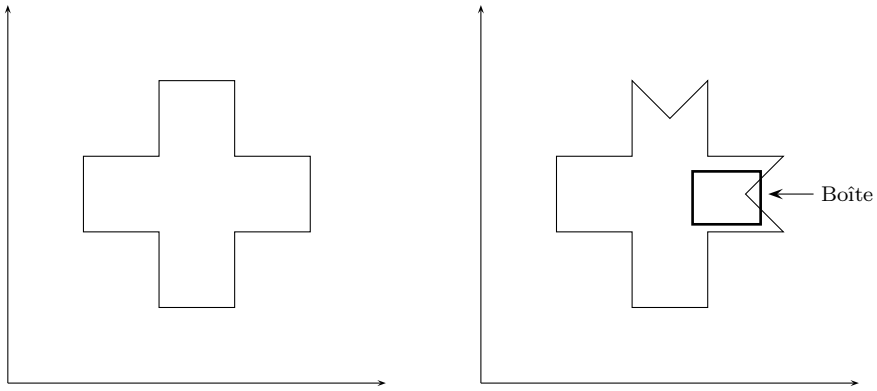
où $Sommets(\overrightarrow{[x]})$ est la fonction qui calcule les points extrêmes (les coins) d'une boîte $\overrightarrow{[x]}$ donnée. La figure 8.3 montre deux exemples d'ensembles : l'ensemble de gauche est axe-convexe, celui de droite ne l'est pas.

À l'instar de la convexité, nous pouvons calculer l'enveloppe axe-convexe d'un ensemble qui n'est pas axe-convexe. La définition découle directement des définitions précédentes :

Définition 24 Enveloppe axe-convexe : *Soit A un ensemble, partie de E. On appelle enveloppe axe-convexe de A et l'on note $EAX(A)$ l'intersection de toutes les parties axe convexes de E contenant A. C'est, au sens de l'inclusion, le plus petit ensemble axe-convexe contenant A.*

La figure 8.4 montre l'enveloppe axe-convexe de l'ensemble non axe-convexe présenté à la figure 8.3.

Proposition 1 *Un ensemble convexe est axe-convexe (mais la réciproque n'est pas vraie)*



(a) Ensemble axe-convexe. (b) Ensemble non axe-convexe.

FIG. 8.3 – *Propriété d'axe-convexité.*

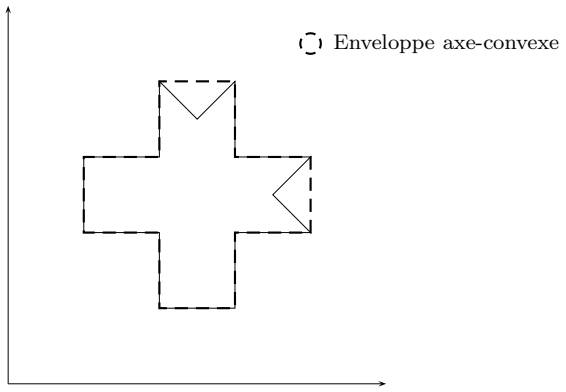
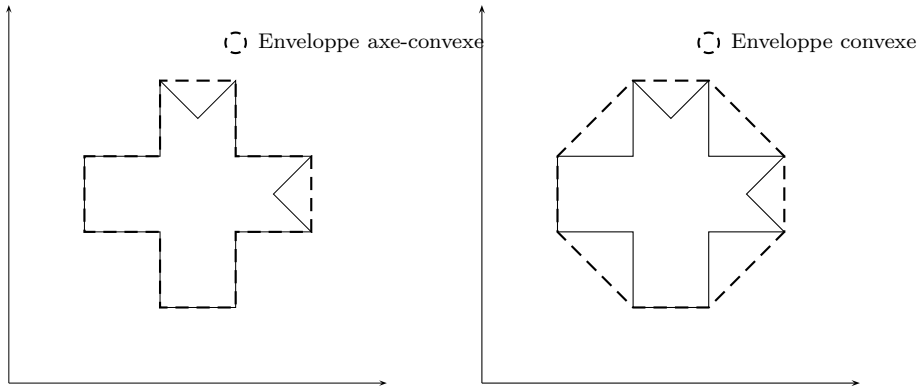


FIG. 8.4 – *Enveloppe axe-convexe.*

La preuve de ce théorème est évidente, en effet, la convexité est un cas particulier de l'axe-convexité. Dès lors si l'ensemble est convexe, il est forcément axe-convexe. Il est clair que l'enveloppe axe-convexe d'un ensemble n'est pas obligatoirement convexe, comme cela est illustré à la figure 8.5.

Proposition 2 *Soit $EAX(A)$ l'enveloppe axe-convexe de A. Si $A \subseteq B$, alors $EAX(A) \subseteq EAX(B)$.*

Preuve :
Soient A et B deux ensembles tels que $A \subseteq B$.



(a) Enveloppe axe-convexe. (b) Enveloppe convexe.

FIG. 8.5 – *Enveloppes convexes et axe-convexes.*

Par définition de l'axe-convexité, $A \subseteq EAX(A)$ et $B \subseteq EAX(B)$.
Donc $A \subseteq B \subseteq EAX(B)$, c'est-à-dire $A \subseteq EAX(B)$.
Or $EAX(B)$ est un ensemble axe-convexe qui contient A.
Par définition, $EAX(A)$ est le plus petit ensemble axe-convexe contenant A, nous en déduisons que $EAX(A) \subseteq EAX(B)$

□

Pour se rapprocher au mieux du cadre des problèmes multiobjectifs continus, nous allons dans la suite de ce chapitre, nous restreindre à $\mathbb{R}^n$. Ainsi, nos définitions et propositions feront apparaître explicitement $\mathbb{R}^n$, en remplacement de l'ensemble E utilisé jusqu'ici.

Définition 25 *Nous appelons régionnement en* $\mathbf{x}$ *toute intersection de n sous-ensembles de* $\mathbb{R}^n$*, chaque sous-ensemble étant construit à partir d'un hyperplan formé à partir des axes du repère orthonormé d'origine* $\mathbf{x}$*. Il existe donc* 2^n *régionnements en* $\mathbf{x}$ *possibles.*

Théorème 2 *Soit A un ensemble fermé de* $\mathbb{R}^n$ *et soit* $\mathbf{x}$ *un point de* $\mathbb{R}^n$ *tel que* $\mathbf{x} \notin A$*. Si A est un ensemble axe-convexe, alors il existe au moins un régionnement en* $\mathbf{x}$ *ne contenant aucun point de A.*

Preuve :

Soit A un ensemble fermé de $\mathbb{R}^n$ et soit $\mathbf{x}$ un point de $\mathbb{R}^n$ tel que $\mathbf{x} \notin A$. Considérons le repère de $\mathbb{R}^n$ d'origine $\mathbf{x}$, il existe 2^n régionnements en $\mathbf{x}$, chacun étant composé à partir de n demi-axes issus de $\mathbf{x}$. Notons (x_i^+) (resp. (x_i^-)) le demi-axe supérieur (resp. inférieur) de la $i^{ème}$ coordonnée de $\mathbf{x}$.

a) Démontrons tout d'abord que : $\forall i \in \{1, \cdots, n\},\ \Big(A \cap (x_i^-) = \emptyset\Big) \vee \Big(A \cap (x_i^+) = \emptyset\Big)$ $\Longrightarrow$ *il existe un régionnement en* $\mathbf{x}$ *ne contenant aucun point de* A.
Supposons : $\forall i \in \{1, \cdots, n\},\ \Big(A \cap (x_i^-) = \emptyset\Big) \vee \Big(A \cap (x_i^+) = \emptyset\Big)$.
Comme $\mathbf{x} \notin A$, il existe au moins n demi-axes $(x_i^{signe_i}),\ signe_i \in \{+, -\}$ ne contenant aucun point de A.
Alors il existe au moins un régionnement en $\mathbf{x}$, appelé $\mathcal{R}$, formé à partir de n demi-axes $((x_0^{signe_0}), (x_1^{signe_1}), \cdots, (x_n^{signe_n}))$.
Comme A est axe-convexe, par définition de l'axe-convexité, il est connexe.
Il en résulte que comme aucun des demi-axes ne contient d'éléments de A, $\mathcal{R}$ ne contient aucun point de A.
Alors il existe un régionnement en $\mathbf{x}$ ne contenant aucun point de A.

b) Démontrons maintenant que : $\exists i \in \{1, \cdots, n\},\ A \cap (x_i^-) \neq \emptyset \wedge A \cap (x_i^+) \neq \emptyset \implies$ *A n'est pas axe-convexe.*
Supposons : $\exists i \in \{1, \cdots, n\},\ A \cap (x_i^-) \neq \emptyset \wedge A \cap (x_i^+) \neq \emptyset$.
Alors, il existe i tel que $\mathbf{a} \in A \cap (x_i^-)$ et $\mathbf{b} \in A \cap (x_i^-)$.
Or $\mathbf{x} \in [\mathbf{a}, \mathbf{b}]$, mais $\mathbf{x} \notin A$, donc $[\mathbf{a}, \mathbf{b}] \not\subset A$.
Comme $[\mathbf{a}, \mathbf{b}]$ est parallèle à un axe du repère, on en déduit que A n'est pas axe-convexe.

c) Nous pouvons maintenant démontrer le théorème.
Prenons la contraposée de l'implication démontrée en **b)** : *A est axe-convexe* $\Longrightarrow$ $\forall i \in \{1, \cdots, n\},\ A \cap (x_i^-) = \emptyset \vee A \cap (x_i^+) = \emptyset$
Grâce à l'implication démontrée en **a)**, nous en déduisons que :
A est axe-convexe $\Longrightarrow$ *il existe un régionnement en* $\mathbf{x}$ *ne contenant aucun point de* A.
□

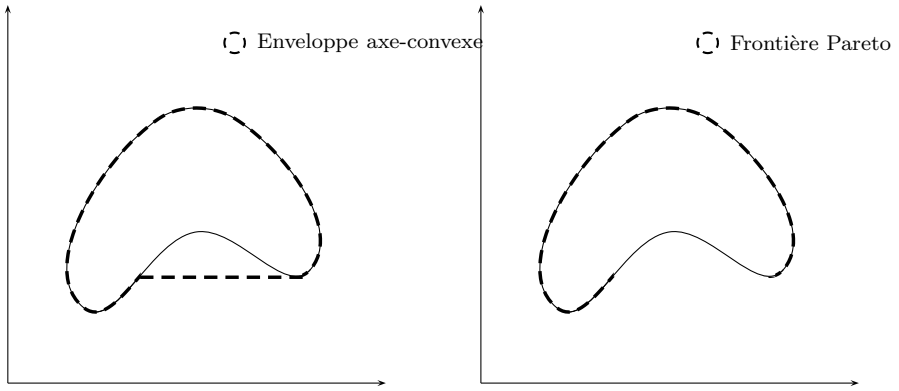
8.2.4 La frontière Pareto

Nous introduisons maintenant un nouveau concept, celui de *la frontière Pareto*. Cette frontière consiste en l'union de plusieurs fronts Pareto (cf. Section 1.5). Nous donnons maintenant la définition formelle de la frontière Pareto :

Définition 26 Frontière Pareto : *Soient* X *un ensemble fermé, et* $\overrightarrow{f} = (f_1, f_2, \cdots, f_m)$ *une fonction à valeurs dans* X. *La frontière Pareto de* X, *notée* $\hat{FP}(X)$, *est définie comme suit :*

$$\hat{FP}(X) = \bigcup_{op \in \{-1,1\}^m} \Big\{ \overrightarrow{x} \in X |\ \forall \overrightarrow{x'} \in X, \Big(\exists i \in [1, \cdots, m]\ op_i \times f_i(\overrightarrow{x}) < op_i \times f_i(\overrightarrow{x'})\Big) \vee \Big(\forall i \in [1, \cdots, m]\ f_i(\overrightarrow{x}) = f_i(\overrightarrow{x'})\Big) \Big\}$$

Un exemple de frontière Pareto se trouve à la figure 8.6. Sur cette figure, la frontière Pareto est constituée à partir des points situés sur le trait en pointillé. Notons que la frontière Pareto n'a donc aucune épaisseur, car elle est constituée à partir de courbes.



(a) Enveloppe axe-convexe. (b) Frontière Pareto.

FIG. 8.6 – *Frontière Pareto.*

La frontière Pareto définie ainsi peut se calculer de plusieurs manières. L'une d'entre elles consiste à utiliser la décomposition en problèmes multiobjectifs. Pour mettre en évidence le lien entre frontière Pareto et problèmes multiobjectifs, il suffit de remarquer que chaque ensemble élémentaire, donné dans la définition, correspond à la formulation d'un problème multiobjectif particulier (cf. Section 1.3). La formulation fait apparaître clairement plusieurs problèmes d'optimisation multiobjectifs à résoudre. Tous les problèmes sont définis à partir des mêmes contraintes (dans la définition, ils utilisent le même ensemble X fermé), mais les objectifs à optimiser sont différents. Ainsi, l'exemple montré à la figure 8.7 fait apparaître quatre problèmes multiobjectifs à résoudre :

- La zone 1 correspond à la résolution du problème où $op = (1, 1)$, c'est-à-dire : min f_1, min $f2$
- La zone 2 correspond à la résolution du problème où $op = (-1, 1)$, c'est-à-dire : max f_1, min $f2$
- La zone 3 correspond à la résolution du problème où $op = (1, -1)$, c'est-à-dire : min f_1, max $f2$
- La zone 4 correspond à la résolution du problème où $op = (-1, -1)$, c'est-à-dire : max f_1, max $f2$

L'union des solutions optimales issues de la résolution de ces quatre problèmes caractérise exactement la frontière Pareto des solutions du problème.

Pour faire le lien avec les précédentes sections portant sur les notions de convexité et d'axe-convexité, nous énonçons le théorème suivant :

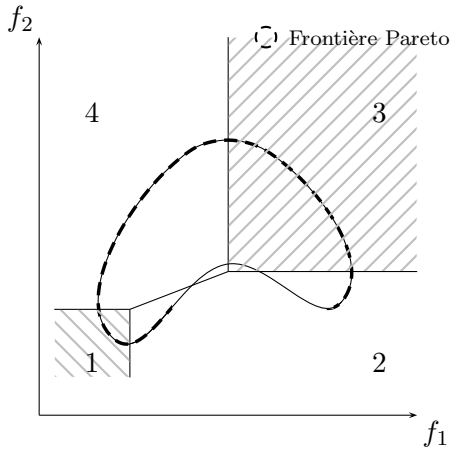


FIG. 8.7 – *Décomposition en problèmes multiobjectifs.*

Théorème 3 *Soit X un ensemble fermé de $\mathbb{R}^n$, alors :*

$$EAX(X) = EAX(\hat{FP}(X))$$

Preuve :

Soit A un ensemble fermé de $\mathbb{R}^n$, et soit $B = \hat{FP}(A)$.
Alors, il est clair que $B \subseteq A$.

a) Démontrons que : $\mathrm{EAX}(B) \subseteq \mathrm{EAX}(A)$.
Comme $B \subseteq A$, d'après la proposition 2, il est clair que $\underline{\mathrm{EAX}(B) \subseteq \mathrm{EAX}(A)}$.

b) Démontrons que : $\mathrm{EAX}(A) \subseteq \mathrm{EAX}(B)$.
Or, d'après la proposition 2, $A \subseteq \mathrm{EAX}(B) \implies \mathrm{EAX}(A) \subseteq \mathrm{EAX}(\mathrm{EAX}(B))$.
Par définition, $\mathrm{EAX}(B)$ est le plus petit ensemble axe-convexe contenant B,
donc $\mathrm{EAX}(\mathrm{EAX}(B)) = \mathrm{EAX}(B)$.
On en déduit que $A \subseteq \mathrm{EAX}(B) \implies \mathrm{EAX}(A) \subseteq \mathrm{EAX}(B)$
Nous cherchons donc à démontrer que $A \subseteq \mathrm{EAX}(B)$, soit que : $\forall \mathbf{x} \in A,\ \mathbf{x} \in \mathrm{EAX}(B)$.
Considérons les deux cas : $\mathbf{x} \in B$ et $\mathbf{x} \in A/B$:

- Soit $\mathbf{x} \in B$, alors par définition de l'axe-convexité, $\mathbf{x} \in \mathrm{EAX}(B)$.
- Soit $\mathbf{x} \in A/B$, $\underline{\text{supposons que } \mathbf{x} \notin \mathrm{EAX}(B)}$.
  Comme $\mathrm{EAX}(B)$ est axe-convexe, d'après le théorème 2, il existe un régionnement $\mathcal{R}$ en $\mathbf{x}$ qui ne contient aucun point de $\mathrm{EAX}(B)$.
  Par définition de $\mathcal{R}$, $\exists op \in \{-1,1\}^n,\ \forall \mathbf{r} \in \mathcal{R},\ \forall i \in \{1,\cdots,n\},\ op_i \times r_i \leq op_i \times x_i$.
  Or, $\mathcal{R}$ ne contient aucun point de B, d'où : $\exists op \in \{-1,1\}^n,\ \nexists \mathbf{y} \in B,\ \forall i \in \{1,\cdots,n\},\ op_i \times y_i \leq op_i \times x_i$.
  Donc, $\exists op \in \{-1,1\}^n,\ \forall \mathbf{y} \in B,\ \exists i \in \{1,\cdots,n\},\ op_i \times x_i < op_i \times y_i$.

Or, $\mathbf{x} \in A$, d'après la définition 26 nous en déduisons que $\mathbf{x} \in \hat{FP}(A)$, donc $\underline{\mathbf{x} \in B}$.
Nous obtenons donc une contradiction car $\mathbf{x} \in A/B$, donc $\underline{x \notin B}$.
Nous en déduisons que notre supposition $x \notin \text{EAX}(B)$ est fausse, donc $\underline{x \in \text{EAX}(B)}$.

D'où, $\forall \mathbf{x} \in A,\ x \in \text{EAX}(B)$, c'est-à-dire $A \subseteq \text{EAX}(B)$.
Comme $A \subseteq \text{EAX}(B) \implies \text{EAX}(A) \subseteq \text{EAX}(B)$, nous en concluons que $\underline{\text{EAX}(A) \subseteq \text{EAX}(B)}$.

c) D'après l'inclusion démontrée en **a)** : $\text{EAX}(B) \subseteq \text{EAX}(A)$.
Et d'après l'inclusion démontrée en **b)** : $\text{EAX}(A) \subseteq \text{EAX}(B)$.
Il est clair que : $\text{EAX}(A) = \text{EAX}(B)$.
Donc : $\underline{\text{EAX}(A) = \text{EAX}(\hat{FP}(A))}$.

□

Ainsi, l'enveloppe axe-convexe d'un ensemble X est égale à l'enveloppe axe-convexe de sa frontière Pareto. La figure 8.6 nous faisait déjà pressentir ce fait, nous en avons maintenant la preuve.

Théorème 4 *Soit X un ensemble fermé de $\mathbb{R}^n$, alors :*

$$\hat{FP}(X) \subseteq \textit{Frontière}(\textit{EAX}(\hat{FP}(X)))$$

où Frontière(X) est la fonction calculant la frontière de X.

Preuve :
Soit A un ensemble fermé de $\mathbb{R}^n$.
$\underline{\text{Supposons que } x \notin \text{Frontière}(\text{EAX}(\hat{FP}(A)))}$.
Comme $\hat{FP}(A) \subseteq \text{EAX}(\hat{FP}(A))$, $x \notin \text{Frontière}(\hat{FP}(A))$.
Or par définition de $\hat{FP}()$, $\hat{FP}(A) = \text{Frontière}(\hat{FP}(A))$.
Donc, $\underline{x \notin \hat{FP}(A)}$.
Nous avons donc : $x \notin \text{Frontière}(\text{EAX}(\hat{FP}(A))) \implies x \notin \hat{FP}(A)$.
En prenant la contraposée de l'implication précédente, nous obtenons :
$x \in \hat{FP}(A) \implies x \in \text{Frontière}(\text{EAX}(\hat{FP}(A)))$.
Donc, $\underline{\hat{FP}(A) \subseteq \text{Frontière}(\text{EAX}(\hat{FP}(A)))}$

□

De par ce théorème, lorsque nous calculons $\hat{FP}(X)$, nous déterminons des parties de la frontière de l'enveloppe axe-convexe de $\hat{FP}(X)$. Or comme, d'après le théorème 3, l'enveloppe axe-convexe de $\hat{FP}(X)$ est égale à l'enveloppe axe-convexe de X, nous en déduisons que calculer $\hat{FP}(X)$ revient à calculer des parties de la frontière de l'enveloppe axe-convexe de X. Or comme tout élément de $\hat{FP}(X)$ est aussi un élément de X, nous calculons en fait des éléments de X faisant partie de la frontière de l'enveloppe axe-convexe de X.

Ainsi, nous approchons de manière très fine X en utilisant ses parties axe-convexes. Nous possédons donc un outil pour approcher les ensembles par leur frontière axe-convexe.

Comme nous ne cherchons à calculer que des parties de la frontière de l'ensemble, nous pouvons espérer des gains d'efficacité par rapport à d'autres méthodes existantes qui cherchent à calculer tout l'ensemble.

8.2.5 La frontière partielle Pareto

Comme nous l'avons vu précédemment, la frontière Pareto fait intervenir plusieurs problèmes multiobjectifs. En effet, si l'espace dans lequel on veut caractériser l'enveloppe est de dimension m, il faudra résoudre 2^m problèmes d'optimisation pour la calculer. Ainsi, cette utilisation reste applicable tant que le nombre de dimensions reste raisonnable. Toutefois, il est possible de définir la notion de frontière partielle Pareto, utilisée pour caractériser certaines parties de la frontière Pareto :

Définition 27 Frontière partielle Pareto : *La frontière partielle Pareto d'un ensemble X fermé, notée $F\hat{P}P(X)$, est définie comme suit :*

$$F\hat{P}P(X) = \bigcup_{op \in \mathcal{OP}} \left\{ \overrightarrow{x} \in X | \, \forall \overrightarrow{x'} \in X, \left(\exists i \in [1, \cdots, m] \; op_i \times f_i(\overrightarrow{x}) < op_i \times f_i(\overrightarrow{x'}) \right) \right. \\ \left. ou \;\; \left(\forall i \in [1, \cdots, m] \; f_i(\overrightarrow{x}) = f_i(\overrightarrow{x'}) \right) \right\}$$

$$Avec \quad \mathcal{OP} \subset \{-1, 1\}^m$$

En reprenant l'exemple donné à la figure 8.7, l'application de cette définition permet de ne calculer que certaines parties de la frontière Pareto, chaque partie étant définie en fonction de l'intersection de la frontière Pareto et d'une des zones 1, 2, 3 ou 4 illustrées sur le schéma. Remarquons que l'union de toutes les frontières partielles Pareto donne exactement la frontière Pareto.

8.2.6 Discussion

Il est clair que les propriétés comme la convexité ou la frontière Pareto ne permettent de calculer qu'une approximation de l'ensemble initial. Dans la plupart des cas, la caractérisation de l'ensemble des solutions dans son intégralité s'avère être une tâche très difficile, alors que le calcul d'une enveloppe convexe ou axe-convexe est plus rapide. Souvent l'approximation de l'ensemble des solutions par son enveloppe axe-convexe est suffisante pour tirer des conclusions sur le problème considéré. Comme nous le verrons dans l'exemple de la prochaine section, certains problèmes ne nécessitent qu'une connaissance partielle de la frontière Pareto, et l'approximation fournie par la frontière partielle Pareto suffit pour que l'utilisateur puisse prendre sa décision au regard des résultats fournis.

8.3 Un exemple d'application : l'analyse de sensibilité

8.3.1 Présentation du problème

L'identification et l'analyse de sensibilité est un domaine d'application en automatique. Le lecteur intéressé trouvera dans le livre de Walter et Pronzato [Walter and Pronzato, 1997] la présentation des notions liées à ce domaine.

Dans les cas traités dans le cadre de cette thèse, nous nous intéressons à l'identification de paramètres incertains d'un système par minimisation d'un critère non convexe. Nous cherchons à visualiser les variations du critère en fonction des zones de confiance des paramètres, dans le but de nous prononcer sur la possibilité d'identifier le système à analyser.

Les différentes instances, ainsi que la définition du critère à minimiser, sont issues de [Dao *et al.*, 2003]. Dans la suite, nous allons traiter deux de ces systèmes, issus du même modèle non linéaire défini ainsi :

$$y_p(t) = g(\overrightarrow{p}, t) = \alpha_1 \exp(-p_1 t) - \alpha_2 \exp(-p_2 t)$$

La figure 8.8 représente la visualisation du système $y_p(t)$ où $\alpha_1 = 20$ et $\alpha_2 = 8$.

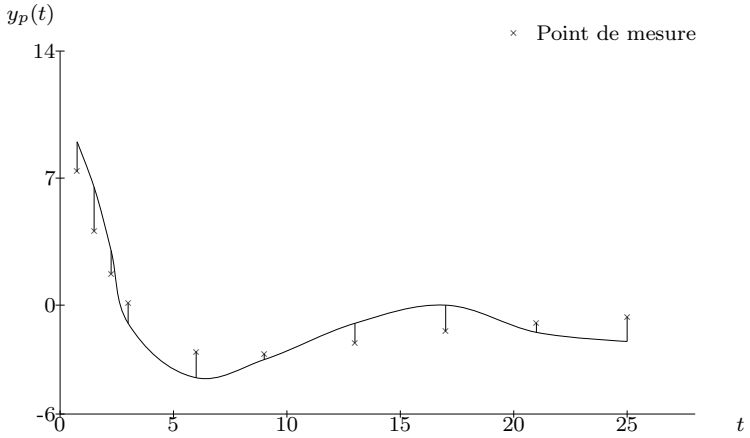


FIG. 8.8 – *Représentation d'un système $y_p(t)$.*

Nous cherchons donc à identifier les paramètres p_1 et p_2 minimisant l'erreur entre le système $y_p(t)$ et les différents points de mesures. Dans [Dao *et al.*, 2003], Dao, Baguenard et Jaulin définissent le critère d'erreur à minimiser de la manière suivante :

$$f(\overrightarrow{p}) = \max_{i=1,\cdots,m} |y_p(t_i) - y_i|$$

Nous cherchons donc à visualiser $f(\overrightarrow{p})$ en fonction de chaque p_k. Nous appelons épigraphe [1] d'un p_k, la partie supérieure de la courbe $f(\overrightarrow{p})$ dans le repère formé à partir des p_k en abscisse et des $f(\overrightarrow{p})$ en ordonnée. Il est clair qu'il est nécessaire de caractériser autant d'épigraphes qu'il y a de critères (deux dans notre cas).

Pour approcher cet épigraphe, nous allons déterminer une frontière partielle Pareto du problème. Le calcul de la frontière Pareto nécessiterait ici la résolution de 4 problèmes multiobjectifs, car le problème de visualisation est en dimension 2. Cependant il est clair que seules les parties de la frontière Pareto incluant les valeurs minimales de $f(\overrightarrow{p})$ sont intéressantes pour notre prise de décision, nous allons donc calculer les frontières partielles Pareto autour de $f(\overrightarrow{p})$.

1. L'épigraphe de f est, par définition, l'ensemble des points du plan situé au dessus de la courbe représentative de f.

8.3.2 Décomposition du problème

Un exemple d'épigraphe est représenté à la figure 8.9. Dans cet exemple, le problème est décomposé en deux problèmes multiobjectifs. Ces deux problèmes sont clairement suffisants pour visualiser les parties intéressantes de l'épigraphe, c'est-à-dire autour des valeurs minimales de $f(\overrightarrow{p})$.

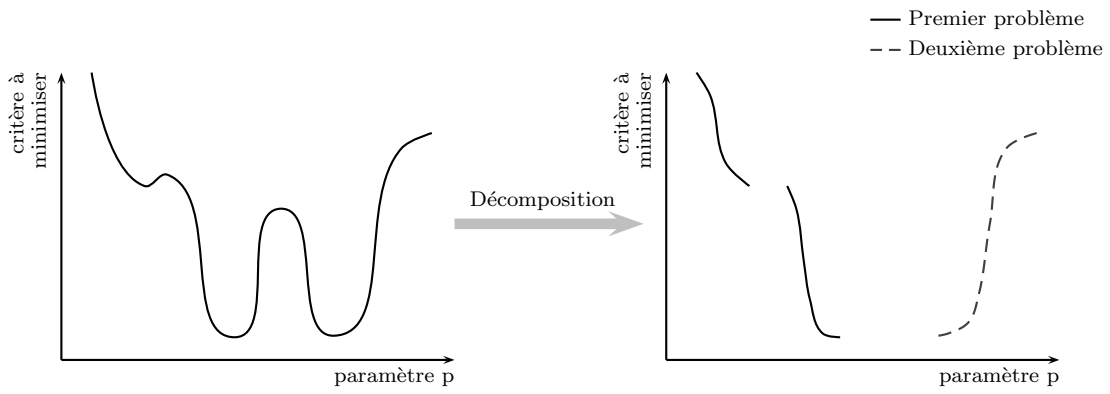


FIG. 8.9 – *Décomposition du problème de visualisation de l'épigraphe.*

Les deux problèmes à résoudre pour caractériser un épigraphe sont définis comme suit :

$$\text{EPI}_k^a \begin{cases} \text{Minimiser} & f_1(x) = \max_{i=1,\cdots,m} |y_p(t_i) - y_i| \\ \text{Minimiser} & f_2(x) = p_k \\ \text{où} & y_i \text{ est la valeur obtenue en mesurant le système à l'instant } t_i \end{cases}$$

$$\text{EPI}_k^b \begin{cases} \text{Minimiser} & f_1(x) = \max_{i=1,\cdots,m} |y_p(t_i) - y_i| \\ \text{Minimiser} & f_2(x) = -p_k \\ \text{où} & y_i \text{ est la valeur obtenue en mesurant le système à l'instant } t_i \end{cases}$$

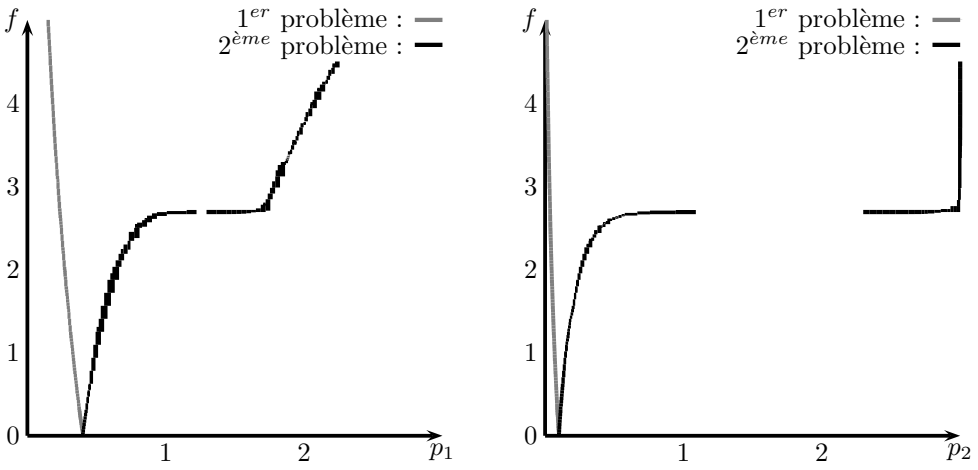
Il est clair, que d'après l'énoncé du problème, il y a deux paramètres p_k à identifier, il est donc nécessaire de visualiser deux épigraphes, ce qui porte à quatre le nombre de problèmes multiobjectifs à résoudre.

8.3.3 Résultats expérimentaux

Nous allons maintenant donner les résultats expérimentaux obtenus par *PICPA-II* sur deux instances de système. Toutes les mesures y_i, $i \in [1, \cdots, m]$ effectuées aux temps t_i sont extraites des travaux de Dao, Baguenard et Jaulin.

Un système facilement identifiable

Pour ce système, nous fixons $\alpha_1 = 20$ et $\alpha_2 = 8$. Le temps de calcul pour la caractérisation d'un épigraphe est inférieur à 10 sec sur un PC (Bi-Pentium III 1 Ghz) sous Linux. La figure 8.10 montre les boîtes calculées par *PICPA-II* pendant la résolution.



(a) Résultat pour le paramètre p_1.

(b) Résultat pour le paramètre p_2.

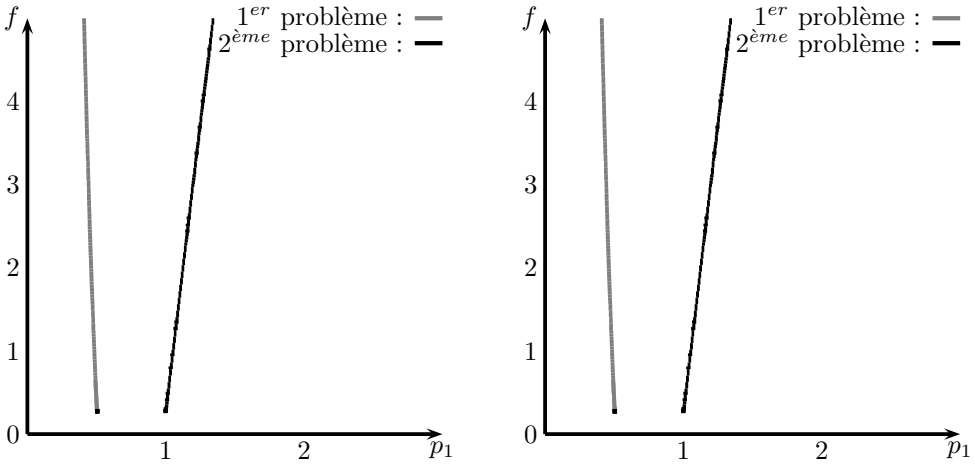
FIG. 8.10 – *Résultats expérimentaux sur un système facilement identifiable.*

Nous ne représentons que les boîtes calculées par *PICPA-II*, car nous sommes assurés que la frontière partielle Pareto calculée est bornée par ces boîtes. Rappelons qu'identifier un système consiste à déterminer la valeur de ses paramètres, ici p_1 et p_2, avec le plus petit taux d'erreur possible. Rappelons aussi que la fonction f calcule l'erreur entre les valeurs théoriques du système et les mesures réelles. Ainsi, lorsque $f = 0$, les valeurs théoriques coïncident parfaitement avec les valeurs réelles. En observant la figure 8.10(a), nous constatons que f prend la valeur 0 en un seul point. L'abscisse de ce point, correspondant à la valeur du paramètre $p1$, est proche de 0.5. Il en va de même pour la figure 8.10(b), où lorsque $f = 0$, le paramètre p_2 est facilement identifiable à une valeur proche de 0.1.

Les deux paramètres p_1 et p_2 sont donc facilement identifiables, car il n'existe pas d'autres choix possibles lorsque f est la plus proche de zéro. Il en résulte que le système est facilement identifiable.

Un système mal identifiable

Pour ce système, nous fixons $\alpha_1 = 100$ et $\alpha_2 = 101$. Le temps de calcul pour la caractérisation d'un épigraphe est inférieur à 10 sec sur un PC (Bi-Pentium III 1 Ghz) sous Linux. La figure 8.11 montre les boîtes calculées par *PICPA-II* pendant la résolution.



(a) Résultat pour le paramètre p_1.

(b) Résultat pour le paramètre p_2.

FIG. 8.11 – *Résultats expérimentaux sur un système mal identifiable.*

Pour ce système aussi, nous ne représentons que les boîtes calculées par *PICPA-II*. En observant la figure 8.11(a), nous constatons premièrement que f, l'écart entre les valeurs théoriques et réelles, ne prend jamais la valeur 0. Sur cette instance, la valeur minimale de f est proche de 0.2. Or, pour cette valeur minimale de f, deux valeurs de p_1 sont possibles. En effet, p_1 peut prendre des valeurs proches de 0.5 et de 1 sans que la valeur de f permette de le différencier. Le paramètre p_1 est donc mal identifiable. Il en va de même pour la figure 8.11(b), car le paramètre p_2 mal identifiable peut prendre plusieurs valeurs différentes (soit une valeur proche de 0.5, soit une valeur proche de 1) pour une même valeur de f proche de 0.2.

Sur ce système, pour une même valeur de f, plusieurs valeurs des paramètres p_1 et p_2 sont possibles, nous sommes donc en présence d'un système mal identifiable. Pour ce système, il est nécessaire de recalibrer les valeurs α_1 et α_2.

8.4 Conclusion

Nous avons montré dans ce chapitre comment utiliser l'optimisation multiobjectif d'une autre manière. Nous avons présenté plusieurs définitions liées à l'axe-convexité et à la frontière Pareto. Nous avons aussi démontré plusieurs théorèmes permettant de lier l'optimisation multiobjectif à l'enveloppe axe-convexe d'un ensemble. Tous ces nouveaux concepts nous ont permis d'utiliser l'optimisation multiobjectif comme outil d'approximation d'un ensemble. En effet, nous avons démontré que l'optimisation multiobjectif était capable de calculer des parties de l'enveloppe axe-convexe d'un ensemble fermé. Grâce à cette enveloppe, nous obtenons une bonne approximation de l'ensemble à approcher.

Cette nouvelle utilisation des problèmes multiobjectifs a trouvé une application dans le

domaine de l'automatique, plus précisément dans l'identification et l'analyse en sensibilité de systèmes. Nous avons utilisé l'algorithme de résolution *PICPA-II* sur deux instances de systèmes distincts. Nous avons montré expérimentalement l'intérêt de *PICPA-II* sur ce type de problème.

Conclusion générale

Résumé des principales contributions

Dans cette thèse, nous avons abordé et étudié les problèmes multiobjectifs sous plusieurs aspects. Ainsi, plusieurs contributions ont été apportées. D'abord, nous avons étudié le problème du sac à dos multidimensionnel multiobjectif et avons développé plusieurs algorithmes d'optimisation différents, afin de mettre en évidence certains aspects de la recherche. À la suite des expérimentations effectuées, nous avons observé l'importance des phases d'*intensification* et de *diversification*. Les méthodes réalisant efficacement les phases d'intensification sont déjà nombreuses. Afin d'enrichir les processus de diversification existants, nous avons développé un processus incrémental de mesure de la diversité, que nous avons ensuite incorporé à un algorithme Tabou. L'algorithme *TS+HW* ainsi obtenu a permis de mettre en évidence la complémentarité des deux phases d'intensification et de diversification. Pour augmenter la puissance de résolution nous avons développé un algorithme hybride combinant une méthode de recherche Tabou et un algorithme évolutionnaire : GTS^{MOKP}. Dans les expérimentations réalisées, nous avons pu observer l'efficacité de notre méthode, en la comparant avec deux algorithmes de référence.

Ensuite, nous nous sommes penchés sur l'hybridation de méthodes exactes et de méthodes approchées, pour obtenir des algorithmes de résolution plus performants. Nous avons considéré cette fois-ci les problèmes d'optimisation continus multiobjectifs sous contraintes. Nous avons ainsi développé *PICPA*, un algorithme intégrant des mécanismes de propagation de contraintes et d'analyse par intervalles avec des concepts évolutionnaires (population et sélection). *PICPA* est un algorithme déterministe calculant des solutions approchées tout en déterminant des bornes du front Pareto. C'est, à notre connaissance, le premier algorithme effectuant chacune de ces deux actions complémentaires, simultanément. En continuant nos travaux, nous avons développé *PICPA-II*, qui améliore *PICPA* sur deux des aspects essentiels : la sélection et le processus d'instanciation. Ainsi, nous avons développé une sélection améliorée éliminant encore plus de points non optimaux, ainsi qu'un processus d'instanciation basé sur une méthode de recherche locale, appelée DLS. La méthode de recherche locale DLS combine la propagation par contraintes et certains concepts issus des stratégies d'évolution pour le calcul d'une solution réalisable. Les deux algorithmes *PICPA* et *PICPA-II* encadrent de manière garantie l'intégralité du front Pareto optimal. Ils calculent de plus des solutions approchées de très bonne qualité, par rapport aux algorithmes de référence, sur un ensemble de problèmes test de la littérature. L'encadrement calculé par *PICPA* et *PICPA-II* permet de bien apprécier la qualité des solutions approchées. En effet, cet encadrement permet de borner l'erreur séparant une solution approchée du front Pareto optimal.

Enfin, nous avons présenté plusieurs définitions liées à l'axe-convexité et à la frontière Pareto. Nous avons aussi démontré plusieurs théorèmes permettant de lier l'optimisation multiobjectif à l'enveloppe axe-convexe d'un ensemble. En effet, nous avons démontré que

l'optimisation multiobjectif permettait de calculer des parties de l'enveloppe axe-convexe d'un ensemble fermé. L'enveloppe ainsi déterminée donne une approximation de cet ensemble fermé. Il est donc possible d'utiliser l'optimisation multiobjectif comme outil d'approximation d'un ensemble. Nous avons illustré cette utilisation en appliquant *PICPA-II* à un problème réel issu de la robotique.

Principales limitations

Ces diverses contributions sont toutefois soumises à certaines limites. En considérant les contributions apportées au problème du sac à dos multidimensionnel multiobjectif, plusieurs points restent encore à approfondir. En effet, bien que très simple, l'algorithme *TS+HW* obtient d'excellents résultats. Ils sont cependant inférieurs à ceux des meilleurs algorithmes actuels beaucoup plus complexes. Nous pensons que le concept adopté par *TS+HW* est adapté à la résolution de problèmes multiobjectifs, mais ne suffit pas à lui seul pour obtenir les meilleurs résultats. Cette mesure fine de la diversité doit être intégrée aux meilleurs méthodes existantes pour améliorer leur aptitude à diversifier. L'algorithme GTS^{MOKP} est un tout premier pas dans ce sens, mais la difficulté pour analyser les résultats d'un algorithme testé sur un grand nombre d'instances de problèmes nous a empêché pour le moment d'aboutir dans cette voie.

Notre algorithme déterministe *PICPA* calcule des bornes des solutions optimales du problème et peut même, dans certains cas, détecter l'inconsistance du problème. Ces propriétés sont un réel atout pour un algorithme, mais bien que *PICPA* nécessite moins de temps de calcul qu'une méthode exacte, ces propriétés le rendent plus sensible à l'augmentation du nombre de variables du problème traité. *PICPA-II* est une réponse à ce problème. En contrepartie, *PICPA-II* perd l'aspect déterministe, mais garde les autres propriétés, notamment la garantie des solutions. La principale limite de *PICPA* et *PICPA-II* concerne le nombre d'objectifs du problème à traiter. Notons que cette limite reste vrai pour les autres méthodes approchées existantes. La quasi totalité des problèmes traités par des algorithmes multiobjectifs sont des problèmes à deux objectifs. Pour obtenir une précision identique lorsque le problème comporte plus de 3 objectifs, il est nécessaire d'augmenter la taille de la population. L'augmentation du nombre d'individus accroît, dans le cas de *PICPA* et *PICPA-II*, le nombre de bissections, et augmente donc d'autant le temps de calcul. Ainsi, lorsque le nombre d'objectifs du problème augmente, il est souvent nécessaire de choisir entre efficacité et précision. Une solution envisageable serait de développer des versions distribuées de *PICPA* et *PICPA-II*. En effet, mis à part la sélection, tous les mécanismes de *PICPA* et *PICPA-II*, tels que la contraction et l'instanciation, ne s'appliquent qu'à un seul individu, et peuvent donc être exécutés de manière indépendante.

Perspectives de recherche

Les premières perspectives de recherche que nous pouvons donner concernent les améliorations de nos algorithmes. Nous avons développé durant cette thèse plusieurs algorithmes dédiés aux problèmes multiobjectifs. Ces algorithmes peuvent bien sûr être améliorés, no-

tamment au niveau de l'efficacité (le temps de calcul de résolution), mais aussi étendus grâce à l'incorporation d'autres heuristiques ou opérateurs de réduction tels que ceux utilisés dans *la programmation linéaire*, *la programmation par contrainte* ou *l'optimisation globale*. Les différents algorithmes que nous avons développés intègrent des opérateurs originaux. Ainsi, *TS+HW* opère avec une heuristique de surveillance de la diversité basée sur la distance de Hamming, et *PICPA-II* comprend un opérateur de recherche local dichotomique, appelé *DLS*. Ces opérateurs peuvent être combinés à d'autres algorithmes, voire même étendus pour devenir des méthodes à part entière. Par exemple, l'opérateur *DLS*, utilisé dans cette thèse pour le calcul d'une solution réalisable, peut être appliqué à des problèmes d'optimisation continus ou des problèmes combinatoires. Cette heuristique hybride, combinant une méthode de stratégie d'évolution, restreinte à une population d'un seul individu, et des opérateurs de propagation de contraintes, peut être vue comme un modèle d'hybridation entre la propagation de contraintes et les méthodes de recherche locale. Comme nous l'avons évoqué précédemment, le développement de versions distribuées de nos algorithmes est une solution viable permettant de traiter des problèmes de plus grande taille et comportant plus d'objectifs. Nous prêterons donc une attention particulière à l'étude de cette solution.

Nous avons aussi vu qu'il était difficile d'évaluer les performances des algorithmes exécutés sur un grand nombre d'instances d'un problème. Il est donc important d'approfondir les travaux existants pour apporter des solutions plus satisfaisantes. Les calculs de bornes effectués par *PICPA* et *PICPA-II* abondent dans ce sens. Il est important d'avoir de meilleurs outils d'estimation de la qualité du front Pareto pour évaluer les performances qualitatives des nombreuses méthodes développées actuellement.

Les différentes définitions et propriétés énoncées au dernier chapitre ouvrent la voie sur une autre utilisation de la modélisation par des problèmes multiobjectifs. Les problèmes multiobjectifs se retrouvent ainsi au cœur de problèmes éloignés des préoccupations initiales de l'optimisation multiobjectif. Dans cette thèse nous avons mis en valeur le lien permettant de passer du domaine de l'optimisation multiobjectif au domaine de l'approximation d'ensemble. Les travaux débutés peuvent être enrichis notamment au travers d'applications ou d'extensions des propriétés avancées.

Index

Liste des publications personnelles

Revues internationales avec comité de rédaction

1. V. Barichard and J.K. Hao. Genetic tabu search for the multi-objective knapsack problem. *Tsinghua Science and Technology*, 8(1):8–13, 2003.

Conférences et ateliers internationaux avec comité de sélection

1. V. Barichard, H. Deleau, J.K. Hao and F. Saubion. A hybrid evoluionary algorithm for CSP. To appear in *Lecture Notes in Computer Science (AE'03)*, France, 2003. Springer

2. V. Barichard and J.K. Hao. A population and interval constraint propagation algorithm. In *Lecture Notes in Computer Science (EMO'03)*, volume 2632, pages 88–101, Faro, Portugal, 2003. Springer.

3. V. Barichard and J.K. Hao. An empirical study of tabu search for the mokp. In Series of Information & Management Sciences, editor, *Proceedings of the First International Workshop on Heuristics*, volume 4, pages 47–56, China, 2002.

4. N. Jussien and V. Barichard. The palm system: explanation-based constraint programming. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming System, a Post-Conference of CP 2000*, pages 118–133, Singapour, 2000.

Conférences nationales avec comité de sélection

1. V. Barichard, H. Deleau, J.K Hao, et F. Saubion. Evolution + adaptation = résolution. Dans *les actes des 12ièmes Journées Francophones de Programmation Logique et programmation par Contraintes (JFPLC'03)*, pages 281–294, Amiens, 2003. Hermès.

2. V. Barichard et J.K Hao. Une approche hybride pour l'optimisation multiobjectif sous contraintes. Dans *les actes des 12ièmes Journées Francophones de Programmation Logique et programmation par Contraintes (JFPLC'03)*, pages 33–48, Amiens, 2003. Hermès.

3. V. Barichard et J.K Hao. Un algorithme hybride pour le sac à dos multiobjectif. Dans *les actes des 8ièmes Journées Nationales sur la Résolution Pratique de Problèmes NP-Complets (JNPC'02)*, pages 19–30, Nice, 2002.

Conférences internationales avec comité de sélection sur résumé

1. V. Barichard and J.K Hao. A population and interval constraint propagation algorithm for mulitobjective optimization. In *Proceedings of The Fifth Metaheuristics International Conference*, pages (06-1)–(06-6), Japan, 2003.

Conférences nationales avec comité de sélection sur résumé

1. V. Barichard et J.K Hao. Picpa : un nouvel algorithme hybride pour les problèmes continus multiobjectifs sous contraintes. Dans *les actes du Cinquième Congrès de la Société Française de recherche Opérationnelle et d'Aide à la Décision (ROADEF'03)*, pages 44–45, Avignon, 2003.

2. V. Barichard et J.K Hao. Étude de la diversité dans un contexte multi-critère. Dans *les actes du Quatrième Congrès de la Société Française de recherche Opérationnelle et d'Aide à la Décision (ROADEF'02)*, pages 44–45, Paris, 2002.

3. V. Barichard, J.K Hao, et F. Saubion. Heuristiques pour la migration et l'évolution des éléments d'un réseau cellulaire. Dans *les actes du Quatrième Congrès de la Société Française de recherche Opérationnelle et d'Aide à la Décision (ROADEF'02)*, pages 46–47, Paris, 2002.

En révision

1. V. Barichard and J.K. Hao. An improved population and interval constraint propagation algorithm. In *IEEE Trans. on Evolutionary Computation*.

2. V. Barichard and J.K. Hao. An empirical study of Tabu search for the 0-1 multidimensional muli-objective knapsack problem. In *Journal of Heuristics*.

Références bibliographiques

[Aarts and Korst, 1989] cité page 25, 25
E.H.L. Aarts and J. Korst. Simulated annealing and boltzmann machines: a stochastic approach to combinatorial and neural computing. *Wiley, Chichester*, 1989.

[Back *et al.*, 1991] cité page 99, 100, 101, 101, 101
T. Back, F. Hoffmeister, and H. Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, 1991.

[Back *et al.*, 1997] cité page 26
T. Back, D.B. Fogel, Z. Michalewicz, and T. Baeck. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

[Barichard and Hao, 2002] cité page 61
V. Barichard and J.K. Hao. An empirical study of tabu search for the mokp. In Series of Information & Management Sciences, editor, *Proceedings of the First International Workshop on Heuristics*, volume 4, pages 47–56, China, 2002.

[Barichard and Hao, 2003a] cité page 65
V. Barichard and J.H. Hao. Genetic tabu search for the multi-objective knapsack problem. *Tsinghua Science and Technology*, 8(1):8–13, 2003.

[Barichard and Hao, 2003b] cité page 97
V. Barichard and J.K. Hao. An improved population and interval constraint propagation algorithm. submitted for publication, 2003.

[Barichard and Hao, 2003c] cité page 79
V. Barichard and J.K. Hao. A population and interval constraint propagation algorithm. In *Lecture Notes in Computer Science (EMO'03)*, volume 2632, pages 88–101, Faro, Portugal, 2003. Springer.

[Benhamou *et al.*, 1999] cité page 83
F. Benhamou, F. Goualard, L. Granvilliers, and J.F. Puget. Revising hull and box consistency. In *Proceedings of the International Conference on Logic Programming*, pages 230–244, 1999.

[Binh and Korn, 1997] cité page 108
T.T. Binh and U. Korn. Mobes: A multiobjective evolution strategy for constrained

optimization problems. In *Proceedings of the Third International Conference on Genetic Algorithms (Mendel 97)*, pages 176–182, 1997.

[Blickle, 1996] cité page 31
T. Blickle. *Theory of evolutionary algorithms and application to system-synthesis*. PhD thesis, Swiss Federal Institute of Technology (Zurich), 1996.

[Bouyssou *et al.*, 2001] cité page 19
D. Bouyssou, E. Jacquet-Lagrèze, P. Perny, R. Slowiński, D. Vanderpooten, and P. Vincke. *Aiding decisions with multiple criteria*. Kluwer, 2001.

[Charnes and Cooper, 1961] cité page 19
A. Charnes and W.W. Cooper. *Management models and industrial applications of linear programming*. Wiley, New York, 1961.

[Cleary, 1987] cité page 80, 81
J.G. Cleary. Logical arithmetic. *Future Computing Systems*, 2(2):125–149, 1987.

[Coello *et al.*, 2002] cité page 31
C. Coello, G. Lamont, and D. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Kluwer, 2002.

[Coello, 1998] cité page 19
C.A. Coello Coello. An updated survey of G.A. based multiobjective optimization techniques. Technical report, Lania-RD-98-08, Xalapa, Veracruz, Mexico, 1998.

[Cohon, 2000] cité page 50
J.L. Cohon. Multiobjective programming and planning. *Mathematics in Science and Engineering, Academic Press, Inc*, 140, 2000.

[Collette and Siarry, 2002] cité page 10, 14, 40, 41
Y. Collette et P. Siarry. *Optimisation multiobjectif*. Eyrolles, 2002.

[Corne and Knowles, 2000] cité page 31, 31, 48, 83
D.W. Corne and J.D. Knowles. M-paes: a memetic algorithm for multiobjective optimization. In *Proceedings of the 2000 Congres on Evolutionary Computation*, pages 325–332, 2000.

[Corne *et al.*, 1999] cité page 22
D. Corne, M. Dorigo, and F. Glover. *New ideas in optimization*. McGraw Hill, 1999.

[Czyzak and Jaszkiewicz, 1998] cité page 31
P. Czyzak and A. Jaszkiewicz. A Pareto simulated annealing - a metaheuristic for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47, 1998.

[Dao *et al.*, 2003] cité page 118, 128, 128
M. Dao, X. Baguenard, et L. Jaulin. Projection d'ensembles pour l'estimation de paramètres, la conception de robot et la commande robuste. *JDA 2003*, pages 4–5, 2003.

[Davis, 1987] cité page 80, 81
E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331, 1987.

[De Jong, 1975] cité page 28, 31
K.A. De Jong. *An analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[Deb and Agrawal, 1995] cité page 91, 107
K. Deb and R.B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[Deb and Goel, 2001] cité page 31, 80, 90, 90, 90, 112
K. Deb and T. Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *Proceedings of Evolutionary Multi-Criterion Optimization*, pages 67–81, 2001.

[Deb and Goyal, 1996] cité page 101
K. Deb and M. Goyal. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.

[Deb *et al.*, 2000] cité page 29, 32, 33
K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI (PPSN-VI)*, pages 849–858, 2000.

[Deb *et al.*, 2001] cité page 29, 91, 93, 107, 109
K. Deb, A. Pratap, and T. Meyarivan. Constrained test problems for multi-objective evolutionary optimization. In *Proceedings of Evolutionary Multi-Criterion Optimization*, pages 284–298, 2001.

[Deb, 2001] cité page 14, 32, 33, 34, 50, 93, 109, 109
K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley, 2001.

[Dréo *et al.*, 2003] cité page 22
J. Dréo, A. Pétrowski, P. Siarry, et E. Taillard. *Métaheuristiques pour l'optimisation*. Eyrolles, 2003.

[Ehrgott and Gandibleux, 2000] cité page 14, 22, 50, 68
M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22:425–460, 2000.

[Ehrgott, 2000] cité page 10, 14
M. Ehrgott. Multicriteria optimization. In *Lecture Notes in Economics and Mathematical Systems*, volume 491. Springer, 2000.

[Fleischer, 2003] cité page 42
M. Fleischer. The measure of Pareto optima. In *Lecture Notes in Computer Science (EMO'03)*, volume 2632, pages 519–533. Springer, 2003.

[Fogel, 2000] cité page 26
D. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence (second edition)*. IEEE Press, 2000.

[Fonseca and Fleming, 1993] cité page 31
C.M. Fonseca and P.J. Fleming. Genetic algorithms for multi-objective optimization: formulation, discussion and generalization. In *Proceedings of The Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[Galinier and Hao, 1999] cité page 65, 66
P. Galinier and J.K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

[Gandibleux *et al.*, 1997] cité page 31, 31, 48
X. Gandibleux, N. Mezdaoui, and A. Freville. A multiobjective tabu search procedure to solve combinatorial optimization problems. In *Lecture Notes in Economics and Mathematical Systems*, volume 455, pages 291–300. Springer, 1997.

[Glover and Laguna, 1997] cité page 25, 25
F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.

[Glover, 1986] cité page 25
F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986.

[Goldberg and Richardson, 1987] cité page 30
D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum, 1987.

[Goldberg, 1989] cité page 26, 28, 31, 32, 50
D.E. Goldberg. Genetic algorithms for search, optimization, and machine learning. *Reading, MA: Addison-Wesley*, 1989.

[Hamda *et al.*, 2002] cité page 90
H. Hamda, O. Roudenko, and M. Schoenauer. Application of a multi-objective evolutionary algorithm to topological optimum design. In *Adaptive Computing in Design and Manufacture*. Springer, 2002.

[Hamida and Schoenauer, 2000] cité page 80
S. B. Hamida and M. Schoenauer. An adaptive algorithm for constrained optimization problems. In *Proceedings of Parallel Problem Solving from Nature VI*, pages 529–538. Springer, 2000.

[Hansen, 1997] cité page 31, 31
M.P. Hansen. Tabu search for multiobjective optimization: Mots. In *Proceedings of 13th International Conference on MCDM*, pages –, 1997.

[Hao *et al.*, 1999] cité page 22
J.K. Hao, P. Galinier, et M. Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'Intelligence Artificielle*, 13(2):283–324, 1999.

[Holland, 1975] cité page 26, 26
J.H. Holland. *Adaptation in natural and artificial systems.* PhD thesis, University of Michigan Press, 1975.

[Hughes, 2003] cité page 90
E.J. Hughes. Multi-objective binary search optimisation. In *Lecture Notes in Computer Science (EMO'03)*, volume 2632, pages 102–117, Faro, Portugal, 2003. Springer.

[Ishibuchi and Murata, 1996] cité page 29
H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm. In *Proceedings of IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 119–124. IEEE, 1996.

[Jaszkiewicz, 2000a] cité page 59, 68, 70
A. Jaszkiewicz. Experiments done with the momhlib: http://www-idss.cs.put.poznan.pl/ jaszkiewicz/momhlib/, 2000.

[Jaszkiewicz, 2000b] cité page 59, 65, 66, 66, 70
A. Jaszkiewicz. On the performance of multiple objective genetic local search on the 0/1 knapsack problem: a comparative experiment. Technical Report RA-002, Research report, Institute of Computing Science, Poznan University of Technology, 2000.

[Jaszkiewicz, 2002] cité page 48, 59, 61
A. Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71, 2002.

[Jaulin *et al.*, 2001] cité page 81, 104
L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics.* Springer-Verlag, London, 2001.

[Jozefowiez *et al.*, 2002] cité page 48
N. Jozefowiez, F. Semet, et E-G. Talbi. Une méta-heuristique parallèle et hybride pour un problème de tournées de véhicules multi-critère. Dans *ROADEF'2002*, 2002.

[Kirkpatrick *et al.*, 1983] cité page 24
S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[Koulamas *et al.*, 1994] cité page 25
C. Koulamas, S.R. Anthony, and R. Jean. A survey of simulated annealing application to operations research problems. *OMEGA*, 22:41–56, 1994.

[Mackworth, 1977] cité page 81, 83
A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[Michalewicz and Schoenauer, 1996] cité page 80
Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[Miettinen, 1999] cité page 14, 17, 50
K. Miettinen. *Nonlinear multiobjective optimization.* Kluwer Academic Publishers, 1999.

[Mohr and Henderson, 1986] cité page 83
R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.

[Moore, 1979] cité page 81, 81
R.E. Moore. Methods and applications of interval analysis. *SIAM, Philadelphia, PA*, 1979.

[Morrison and De Jong, 2001] cité page 53
W. Morrison and K.A De Jong. Measurement of population diversity. In *Artificial Evolution*, pages 31–41. Springer, 2001.

[Morse, 1980] cité page 36
J.N. Morse. Reducing the size of the nondominated set: pruning by clustering. *Computers and Operations Research*, 7:55–66, 1980.

[Osyczka and Kundu, 1995] cité page 92
A. Osyczka and S. Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, 10:94–99, 1995.

[Othmani, 1998] cité page 10
I. Othmani. *Optimisation multicritère : Fondements et Concepts.* PhD thesis, Université de Grenoble, 1998.

[Parks and Miller, 1998] cité page 29
G.T. Parks and I. Miller. Selective breeding in a multiobjective genetic algorithm. In *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V)*, pages 250–259. Springer, 1998.

[Pirlot and Teghem, 2002] cité page 22
M. Pirlot and J. Teghem. *Optimisation approchée en recherche opérationnelle (Traité IC2, Série Informatique et systèmes d'information).* Hermès Sciences, 2002.

[Pirlot and Teghem, 2003] cité page 22
M. Pirlot and J. Teghem. *Résolution de problèmes de RO par les métaheuristiques (Traité IC2, série Informatique et systèmes d'information).* Hermès Sciences, 2003.

[Reeves, 1995] cité page 22
C.R. Reeves. *Modern heuristic techniques for combinatorial problems.* McGraw Hill, 1995.

[Romero, 1991] cité page 19
C. Romero. *Handbook of critical issues in goal programming.* Oxford, UK: Pergamon Press, 1991.

[Schaffer, 1984] cité page 48
J.D. Schaffer. *Multiple objective optimization with vector evaluated genetic algorithms.* PhD thesis, Vanderbilt University, 1984.

[Schott, 1995] cité page 40
J.R. Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1995.

[Schütze *et al.*, 2003] cité page 90
O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. Covering Pareto sets by multilevel evolutionary subdivision techniques. In *Lecture Notes in Computer Science (EMO'03)*, volume 2632, pages 118–132, Faro, Portugal, 2003. Springer.

[Schwefel, 1981] cité page 26
H-P. Schwefel. *Numerical optimization of computer models.* Wiley, Chichester, 1981.

[Selman *et al.*, 1994] cité page 51
Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *AAAI*, volume 1, pages 337–343, 1994.

[Serafini, 1992] cité page 48
P. Serafini. Simulated annealing for multiobjective optimization problems. In *10th Int. Conf. on MCDM*, pages 87–96, 1992.

[Srinivas and Deb, 1994] cité page 28, 31, 32, 48, 83
N. Srinivas and K. Deb. Multiobjective optimization using non dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[Talbi, 2000] cité page 31
E-G. Talbi. Une taxinomie des métaheuristiques hybrides. Dans *ROADEF'2000*, 2000.

[Tamaki *et al.*, 1994] cité page 29
H. Tamaki, M. Mori, M. Araki, Y. Mishima, and H. Ogai. Multicriteria optimization by genetic algorithms: A case of scheduling in hot rolling process. In *Proceedings of APORS'94*, pages 374–381. World Scientific Publishing, 1994.

[Tanaka, 1995] cité page 91
M. Tanaka. GA-based decision support system for multi-criteria optimization. In *Proceedings of the International Conference on Systems, Man and Cybernetics-2*, pages 1556–1561, 1995.

[Tsang, 1993] cité page 99
E. Tsang. Foundations of constraint satisfaction. *Academic Press, London*, 1993.

[Ulungu and Teghem, 1994a] cité page 14
E.L. Ulungu and J. Teghem. Multi-objective combinatorial optimization: a survey. *Journal of Multi-Criteria Decision Analysis*, 3:83–104, 1994.

[Ulungu and Teghem, 1994b] cité page 22
E.L. Ulungu and J. Teghem. The two-phases method: an efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2):149–165, 1994.

[Ulungu *et al.*, 1999] cité page 48
E.L. Ulungu, J. Teghem, Ph. Fortemps, and D. Tuyttens. MOSA method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8:221–336, 1999.

[Ulungu, 1993] cité page 22
E.L. Ulungu. *Optimisation combinatoire multicritère : détermination de l'ensemble des solutions efficaces et méthodes interactives*. PhD thesis, Faculté des Sciences, Université de Mons-Hainault, 1993.

[Vasquez and Hao, 2001] cité page 61, 76
M. Vasquez and J.K. Hao. Une approche hybride pour le problème de sac à dos multi-dimensionnel. *RAIRO Operational Research*, 35(4):415–438, 2001.

[Veldhuizen, 1999] cité page 42
D.A.V Veldhuizen. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. PhD thesis, Graduate School of Engineering of the Air Force Institute of Technology, Air University, 1999.

[Vidal, 1993] cité page 25
R.V. Vidal. Applied simulated annealing. *Lecture Notes in Economics and Mathematical Systems*, 396, 1993. Springer-Verlag.

[Vincke, 1992] cité page 19
Ph. Vincke. *Multicriteria decision aid*. J. Wiley, New York, 1992.

[Visée *et al.*, 1998] cité page 22
M. Visée, J. Teghem, M. Pirlot, and E.L. Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12:139–155, 1998.

[Walter and Pronzato, 1997] cité page 127
E. Walter and L. Pronzato. *Identification of parametric models from experimental data*. Springer-Verlag, 1997.

[Zitzler and Thiele, 1998a] cité page 48
E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization: the stength Pareto approach. Technical report, Swiss Federal Institute of Technology (Zurich), 1998.

[Zitzler and Thiele, 1998b] cité page 34
E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms: a comparative case study. In *Lecture Notes in Computer Science*, pages 292–301. Springer, 1998.

[Zitzler and Thiele, 1999] cité page 29, 29, 31, 31, 54, 56, 59, 59, 70, 72, 83, 112
E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3:257–271, 1999.

[Zitzler *et al.*, 2000] cité page 31, 111
E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation Journal*, 8(2):125–148, 2000.

[Zitzler, 1999] cité page 35, 41, 43
E. Zitzler. *Evolutionary algorithms for multiobjective optimization: methods and applications*. PhD thesis, Swiss Federal Institute of Technology (Zurich), 1999.

Approches hybrides pour les problèmes multiobjectifs

Résumé

Cette thèse porte sur la résolution des problèmes d'optimisation multiobjectifs. Les contributions apportées sont de trois types : premièrement, nous développons pour le problème du sac à dos multidimensionnel multiobjectif des algorithmes originaux fondés sur la recherche Tabou ou évolutionnaire intégrant une gestion efficace de la diversité. Nous mettons ainsi en évidence l'importance de la diversité pour ce problème. Deuxièmement, nous développons pour les problèmes multiobjectifs continus sous contraintes une méthode originale fondée sur une représentation par intervalles, des méthodes de propagation de contraintes, des algorithmes de recherche locale et des concepts évolutionnaires. Notre méthode est capable de fournir à la fois des bornes et des solutions approchées du front Pareto.

Mots-clés : optimisation, problèmes multiobjectifs, algorithmes évolutionnaires, recherche locale, approches complètes, approches hybrides

Hybrid approaches for multiobjective problems

Abstract

This thesis deals with the resolution of multiobjective optimization problems. Our contributions are of three types: first, we develop for the multiobjective muldimensionnal knapsack problem some original algorithms based on Tabu search or evolutionary algorithms. Through out the experiments, we show the importance of diversity and of its efficient management during the search. Subsequently, we develop for multiobjective continuous problems with constraints an original method which is mainly based on a representation with intervals, constraint propagation methods, local search processes and evolutionary concepts.

Keywords: optimization, multiobjective problems, evolutionary algorithms, local search, complete approaches, hybrid approaches