



Méthode générique pour l'optimisation d'agencement géométrique et fonctionnel

Guillaume Jacquenot

► To cite this version:

Guillaume Jacquenot. Méthode générique pour l'optimisation d'agencement géométrique et fonctionnel. Mécanique [physics.med-ph]. Ecole Centrale de Nantes (ECN) (ECN) (ECN) (ECN), 2010. Français.

HAL Id: tel-00468463

<https://tel.archives-ouvertes.fr/tel-00468463>

Submitted on 30 Mar 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

École Centrale de Nantes

École Doctorale
Sciences Pour l'Ingénieur, Géosciences, Architecture

Année 2010

N° B.U. :

Thèse de Doctorat
Diplôme délivré par l'École Centrale de Nantes

Spécialité : GÉNIE MÉCANIQUE

Présentée et soutenue publiquement par :

GUILLAUME JACQUENOT

le 18 janvier 2010

à l'École Centrale de Nantes

Méthode générique pour l'optimisation d'agencement géométrique et fonctionnel

JURY

Président :	Georges FADEL	<i>Professeur, Clemson University, États-Unis d'Amérique</i>
Rapporteurs :	Georges FADEL	<i>Professeur, Clemson University, États-Unis d'Amérique</i>
	Jean-Pierre NADEAU	<i>Professeur, TRÈFLE, ENSAM de Bordeaux</i>
Examineurs :	Carlo POLONI	<i>Professeur, Università degli Studi di Trieste, Italie</i>
	Fouad BENNIS	<i>Professeur, IRCCyN, École Centrale de Nantes, Nantes</i>
	Philippe WENGER	<i>Directeur de recherche CNRS, IRCCyN, Nantes</i>
	Jean-Jacques MAISONNEUVE	<i>Docteur, SIREHNA, Nantes</i>
Invité :	Vitali TELERMAN	<i>Docteur, Dassault Systèmes, Vélizy-Villacoublay</i>

Directeur de thèse :	Fouad BENNIS
Co-directeur de thèse :	Philippe WENGER
Laboratoire :	Institut de Recherche en Communications et Cybernétique de Nantes. UMR CNRS 6597
Encadrant industriel :	Jean-Jacques MAISONNEUVE
Entreprise :	SIREHNA

N° É.D. : 498-93

Remerciements

Je tiens en tout premier lieu à remercier mes encadrants Messieurs Fouad Bennis, Professeur à l'École Centrale de Nantes, Philippe Wenger, Directeur de recherche *CNRS*, ainsi que Jean-Jacques Maisonneuve, docteur dans l'entreprise *SIREHNA*. Leur disponibilité et leur grande ouverture d'esprit m'ont permis de réaliser cette thèse dans les meilleures conditions.

Je tiens à exprimer toute ma gratitude à Jean-Pierre Le Goff, initiateur du projet de recherche sur l'agencement à *SIREHNA*. Je remercie aussi l'*Association Nationale de la Recherche et de la Technologie (ANRT)* ainsi que les financeurs du projet *Multi Disciplinary Optimization* du pôle de compétitivité *Ensembles Métalliques et Composites Complexes*, qui ont participé au financement de ma thèse.

Mes remerciements s'adressent également à Jean-François Lafay et Michel Malabre pour m'avoir accueilli de 2006 à 2010 à l'*Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN)*.

J'adresse également mes remerciements aux membres de mon jury de thèse : je remercie les professeurs George Fadel et Jean-Pierre Nadeau d'avoir accepté d'être rapporteurs de ma thèse. J'ai apprécié l'attention avec laquelle ils ont lu et évalué ce mémoire. Je remercie George Fadel pour avoir accepté de présider mon jury de thèse, ainsi que pour l'intérêt qu'il a porté à mon travail lors de nos différentes rencontres. Je tiens aussi à remercier le professeur Carlo Poloni et le docteur Vitali Telerman qui m'ont fait l'honneur de compléter ce jury.

Enfin, j'adresse mes vifs remerciements à mes collègues de *SIREHNA*, de l'*IRCCyN* et du *LINA*, qui m'ont apporté un soutien technique de grande qualité. J'adresse également mes remerciements à ma famille, qui m'a encouragé et a su être patiente.

Sommaire

Sommaire	v
Nomenclature	vii
Introduction	1
1 Problèmes de placement : définitions et classifications	3
2 Modélisations et algorithmes d’optimisation pour les problèmes de placement	25
3 Optimisation multi-objectif d’agencement de locaux	43
4 Méthode de placement proposée	59
5 Résolution de problèmes de placement bidimensionnels.....	101
Conclusions et perspectives	137
A Abréviations	141
B Outils pour la résolution des problèmes de placement	145
C Résultats annexes.....	163
D Présentation du démonstrateur développé.....	169
 Bibliographie	 171
Liste des tableaux	187
Liste des figures	189
Liste des algorithmes	193
Table des matières.....	195

Nomenclature

Notation	Signification
<i>Problème d'optimisation</i>	
n	Nombre de variables total du problème d'optimisation $n = n_{\text{real}} + n_{\text{disc}} + n_{\text{perm}}$
n_{real}	Nombre de variables réelles
n_{disc}	Nombre de variables discrètes
n_{perm}	Nombre de variables sous forme de permutation
p	Nombre d'objectifs du problème
$\mathbf{f} = (f_1, f_2, \dots, f_p)$	Vecteur des objectifs du problème
<i>Méthode de placement et Algorithme de séparation</i>	
d	Dimension du problème $d \in \{0, 1, 2, 3\}$
m	Nombre de composants
C	Contenant
C_i	Composant i ($1 \leq i \leq m$)
F	Fonction traduisant le non-respect des contraintes de placement
ω_{pen}	Poids d'agrégation des contraintes de non-chevauchement
ω_{pro}	Poids d'agrégation des contraintes d'appartenance
\mathbf{v}	Vecteur des coordonnées des composants
\mathbf{v}_i	Vecteur des coordonnées du composant i ($\mathbf{v}_i = (\mathbf{x}_i, \mathbf{o}_i)$)
\mathbf{x}_i	Vecteur de translation du composant i
\mathbf{o}_i	Vecteur d'orientation du composant i
x_i, y_i, θ_i	Coordonnées du composant i en 2D
$x_i, y_i, z_i, \phi_i, \theta_i, \psi_i$	Coordonnées du composant i en 3D
S_{ij}	Cercle / Sphère j du composant i
\mathbf{c}_{ij}	Coordonnées du cercle / sphère j du composant i
$p_{ij,x}, p_{ij,y}, p_{ij,z}$	Positions relatives du centre de la sphère j par rapport au point de référence du composant i
$\delta(C_i, C_j)$	Profondeur de pénétration entre les composants C_i et C_j
$\delta(S_{ij}, S_{kl})$	Profondeur de pénétration la $j^{\text{ème}}$ sphère du composant C_i avec et la $l^{\text{ème}}$ sphère du composant C_k
<i>Algorithme génétique</i>	
p_c	Probabilité de croisement sur les variables réelles
p_m	Probabilité de mutation sur les variables réelles
p_s	Probabilité d'échange
CV	Indice de violation de contraintes
<i>Indicateurs multi-objectifs</i>	
\mathcal{D}	Distance entre la surface de compromis et le front de Pareto
\mathcal{R}	Mesure de la représentation du front de Pareto
\mathcal{S}	Métrique d'espacement
HV, HVR	Hypervolume, Hypervolume relatif
\mathcal{C}	Métrique de couverture d'ensemble

Introduction

Objectifs de la thèse

Connus et identifiés depuis bien longtemps, les problèmes d'agencement présentent des applications dans tous les domaines industriels. Souvent résolus à la main à partir de l'intuition et l'expérience des concepteurs, le développement de méthodes de résolution informatique devient un enjeu de poids à l'heure où les systèmes deviennent de plus en plus complexes et compacts.

Alors que les solutions technologiques pour l'optimisation de produits sont en plein essor avec l'optimisation multi-objectif et multidisciplinaire, le développement d'outils pour l'agencement met du temps à émerger. Ceci s'explique d'une part par la difficulté de formalisation et de modélisation de ces problèmes, d'autre part par la difficulté à identifier des stratégies de résolution. La maturité des outils de traitement géométrique ainsi que les développements récents de techniques d'optimisation robuste ont relancé l'intérêt porté à ces problèmes.

Qu'ils soient purement géométriques, ou incluant des critères fonctionnels, ces problèmes ont fait l'objet de nombreux travaux dans la littérature. Toutefois, les méthodes de résolution généralement proposées sont spécifiques et ne peuvent être appliquées à différents problèmes. De plus, la majorité d'entre elles traite les problèmes de manière mono-objectif, alors que les problèmes comportent souvent plusieurs objectifs. Les enjeux et la complexité de ces problèmes en font un défi fort intéressant.

Après la réalisation d'un état de l'art et d'une classification des problèmes d'agencement, l'objectif principal de la thèse est de proposer un cadre de travail ouvert pour la résolution générique des problèmes 2D et 3D permettant l'intégration de nouveaux objectifs et contraintes.

Cadre de la thèse

Cette thèse CIFRE s'inscrit dans le cadre du projet *EMC2-MDO*¹, labellisé par le pôle de compétitivité *EMC2*². Ce projet vise à diffuser les technologies d'optimisation de conception multidisciplinaire dans le milieu industriel. Financé par la région Pays de la Loire et le Ministère de l'Industrie, ce projet regroupe les entreprises *SIREHNA*³, *DCNS*⁴, *MecaChrome*⁵, la *Société des Polymères Barre Thomas*⁶, ainsi que l'*École Centrale de Nantes*⁷ et l'*Institut de Recherche en Communications et Cybernétique de Nantes*⁸. Cette thèse s'inscrit plus particulièrement dans une action de recherche plus générique avec *SIREHNA*, l'*ECN* et *IRCCyN*, concernant la problématique de l'optimisation d'agencement, pour laquelle les technologies d'optimisation de conception actuelles atteignent souvent leurs limites.

1. Ensembles Métalliques et Composites Complexes – Multidisciplinary Design Optimization

2. www.pole-emc2.fr

3. www.sirehna.com

4. www.dcnsgroup.com

5. www.mecachrome.com

6. www.barrethomas.com

7. www.ec-nantes.fr

8. www.irccyn.ec-nantes.fr

Organisation du mémoire

La suite de ce mémoire s'articule de la manière suivante. Le chapitre 1 présente un état de l'art et une classification des problèmes d'agencement. Leur analyse nous amène à renommer les problèmes d'agencement en problèmes de placement. Nous verrons que ce terme plus générique regroupe deux familles : les problèmes de découpe et de conditionnement ainsi que les problèmes d'agencement. Les définitions et différents éléments de ces problèmes sont présentés de manière à les formaliser. Les difficultés de formulation, modélisation et résolution, qui motivent ce travail de recherche, sont discutées et analysées. La présentation des différents problèmes et de leurs spécificités donnera une vision globale de la problématique.

Les méthodes de résolution et d'optimisation des problèmes de placement sont présentées dans le chapitre 2. Les problèmes de placement étant souvent multi-objectifs, une attention particulière sera portée à leur modélisation : les aspects théoriques de l'optimisation multi-objectif de même que les algorithmes d'optimisation multi-objectif utilisés dans les chapitres suivants sont présentés. Enfin, les différents outils d'analyse multi-objectif, nécessaires à la validation des résultats présentés, sont introduits. Ces outils permettront par la suite de comparer les résultats d'optimisation.

Le chapitre 3 présente une méthode d'agencement, où la géométrie des composants n'intervient pas directement. Cette méthode est appliquée sur un exemple d'agencement de compartiments d'un navire, issu de la littérature. Modélisé comme un problème d'optimisation combinatoire, la résolution repose sur une méthode de placement originale garantissant le respect des contraintes de placement. Nous proposons une généralisation multi-objectif de la méthode de résolution.

Le chapitre 4 présente l'approche générique proposée pour la résolution de problèmes de placement avec géométrie. Avant de présenter cette approche, les justifications qui nous ont conduits à ce choix sont énoncées. La méthode proposée est une approche hybride basée sur un algorithme évolutionnaire permettant une recherche globale de solutions et un algorithme de séparation garantissant au maximum la satisfaction des contraintes géométriques. Alors que le choix de l'algorithme évolutionnaire est indépendant du problème, l'algorithme de séparation est directement relié à la géométrie des composants. Les différentes modélisations des problèmes ainsi que les différentes variantes de l'algorithme de séparation, que nous avons développées, sont présentées. Plusieurs exemples permettent de visualiser les étapes de l'algorithme de séparation. Les différentes possibilités d'extension de la méthode, qui assurent son caractère générique, sont listées.

Enfin, le chapitre 5 teste la méthode proposée sur plusieurs exemples d'application avec des études détaillées sur les résultats obtenus. Issu de la littérature, le premier exemple permet de comparer la méthode proposée avec des méthodes de référence pour un problème spécifique. Par la suite, deux exemples d'agencement multi-objectifs sont présentés et résolus. Ils permettent de mettre en évidence les points forts de la méthode proposée. Différentes études sont menées indépendamment pour comprendre le rôle des différents composants de la méthode de résolution. L'analyse de la diversité des solutions dans l'espace des variables et des objectifs fait l'objet d'une attention particulière. Les comparaisons entre optimisations sont effectuées à l'aide des indicateurs multi-objectifs présentés dans le chapitre 2. Les différentes optimisations effectuées permettent de proposer des recommandations quant aux réglages des algorithmes.

Chapitre 1

Problèmes de placement : définitions et classifications

Ce chapitre présente un état de l'art ainsi qu'une classification des problèmes de placement. Chaque problème type est analysé et les méthodes de modélisation et de résolution sont décrites.

Sommaire

1.1	Introduction	3
1.2	Formulation des problèmes de placement	4
1.2.1	Définitions générales	4
1.2.2	Définition des problèmes de placement	5
1.2.3	Vocabulaire des problèmes de placement	5
1.3	Caractéristiques des problèmes de placement	5
1.3.1	Techniques de placement	5
1.3.2	Représentation des variables	6
1.3.3	Les objectifs des problèmes de placement	8
1.3.4	Les contraintes des problèmes de placement	8
1.3.5	Évaluation des contraintes de non-chevauchement et d'appartenance	8
1.4	Les problèmes de découpe et de conditionnement	9
1.4.1	Définition des problèmes de découpe et de conditionnement	9
1.4.2	Typologies des problèmes de découpe et de conditionnement	10
1.4.3	Problèmes de découpe	12
1.4.4	Problèmes de sac à dos	12
1.4.5	Le problème de <i>bin packing</i> à deux dimensions	13
1.4.6	Problème de chargement de palettes	15
1.4.7	Problème de découpe de formes irrégulières	15
1.4.8	Problèmes de conditionnement 3D d'objets de géométries quelconques	15
1.5	Les problèmes d'agencement	16
1.5.1	Problèmes d'agencement bidimensionnel	17
1.5.2	Problèmes d'agencement tridimensionnel	21
1.6	Comparaison des problèmes C&P et des problèmes d'agencement	22
1.7	Conseils pour la résolution d'un problème de placement	22
1.8	Conclusion	24

1.1 Introduction

Les problèmes d'agencement géométrique et fonctionnel, ci-après dénommés *problèmes de placement*, regroupent une large gamme de problèmes, à savoir les problèmes de découpe et de conditionnement et les problèmes d'agencement. Les traductions anglophones de ces termes sont respectivement *Cutting & Packing problems (C&P)* et *Layout problems*. Ces problèmes ont fait l'objet de nombreuses études dans la littérature et ont généré une grande quantité de publications. Les raisons à cela sont multiples. Tout d'abord, il existe de nombreuses variantes à ces problèmes, qui peuvent générer des modélisations et des méthodes de résolution différentes. Ensuite, les objectifs et contraintes peuvent varier, nécessitant des adaptations, voire des refontes complètes des modèles. Récemment, l'apparition des méthodes de résolution multi-objectifs a ouvert de nouvelles perspectives permettant la prise en compte de nouveaux aspects. Enfin, les enjeux industriels derrière toutes ces problématiques sont importants, entraînant une recherche perpétuelle sur ces sujets connus depuis de nombreuses années. Les enjeux industriels et financiers ont été les moteurs initiaux de cette recherche. Récemment, les enjeux environnementaux ont apporté une raison supplémentaire à l'intérêt scientifique porté à ces problèmes. La recherche s'inscrit désormais dans une démarche de développement durable. Des exemples peuvent être trouvés pour chacun de ces problèmes de placement.

Les figures 1.1 et 1.2 présentent quelques-uns des problèmes de découpe et de conditionnement ainsi que des problèmes d'agencement. Chaque image est représentative d'un problème donné. L'optimisation de découpe de matières premières (tôles, tissus,...) permet de limiter les coûts de découpe ainsi que les chutes générées pour la production de biens manufacturés. L'optimisation de chargement d'un camion, navire, ou avion permet de maximiser le chargement. De plus, la répartition optimale du chargement permet une usure régulière des pièces mécaniques et des consommables. L'agencement optimal d'un assemblage de composants électroniques permet de limiter son échauffement ainsi que sa consommation électrique. Dans les usines, une disposition optimale des moyens de production permet d'augmenter la productivité, de limiter les flux et les coûts de déplacement.

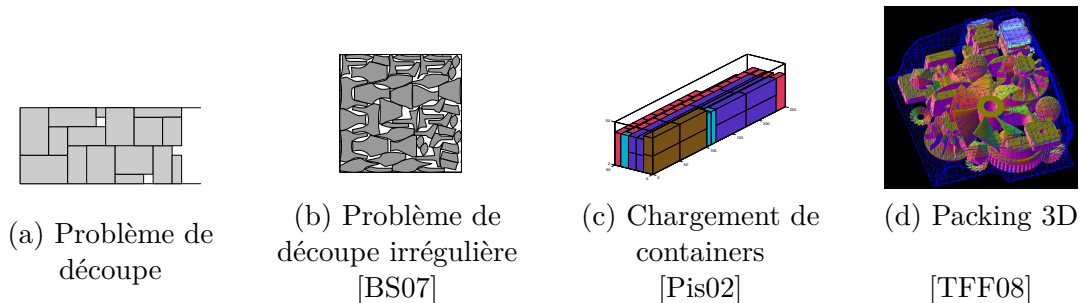


Figure 1.1 – Présentation de quelques-uns des problèmes de découpe et de conditionnement.

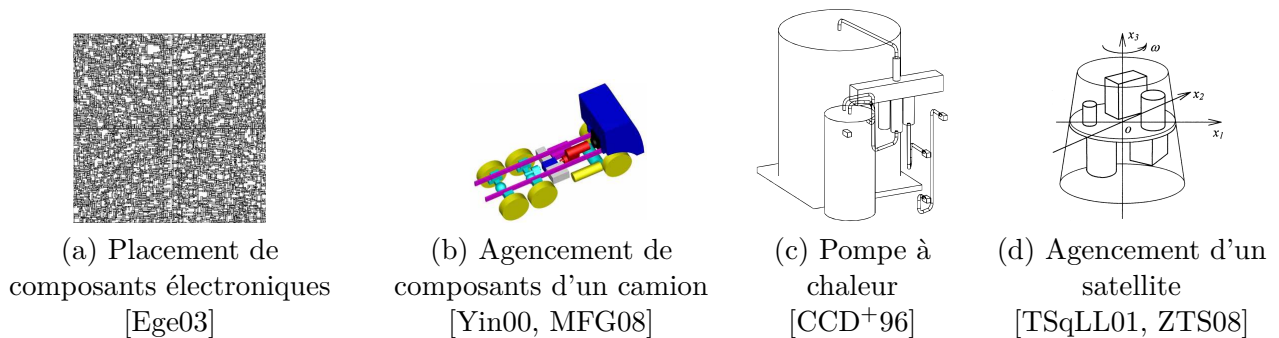


Figure 1.2 – Présentation de quelques-uns des problèmes d'agencement.

1.2 Formulation des problèmes de placement

Tout d'abord, les différents éléments nécessaires à la présentation du problème de placement sont introduits. Ceux-ci permettront par la suite de définir un problème de placement. Ensuite, les différents termes utilisés dans la littérature seront comparés.

1.2.1 Définitions générales

Avant de définir mathématiquement les problèmes de placement en détail, les différents éléments les constituant doivent être définis, à savoir composants, attributs, variables et objectifs et contraintes.

Généralement, un composant est défini comme un élément à positionner caractérisé par une liste d'attributs. Dans le cadre de notre travail, les composants sont le plus souvent des objets physiques. Dans la littérature, certains auteurs considèrent aussi des objets abstraits, présentant une analogie directe avec les problèmes de placement [Dyc90]. Les attributs expriment les propriétés des composants, ils sont utilisés pour évaluer les solutions de placement. Ils peuvent être constants ou variables. Dans la majorité des problèmes rencontrés, ce sont des données qui ne changent pas lors de la résolution. Les composants sont entièrement définis par leur géométrie et leurs attributs. La géométrie d'un composant peut être simplement une longueur pour un problème 1D, ou bien un modèle CAO complet pour les problèmes 3D.

Un contenant est un élément physique défini avec plusieurs attributs, dans lequel l'ensemble des composants doit être positionné. Le contenant peut être de dimensions fixes ou variables. Dans ce dernier cas, l'objectif du problème consiste à minimiser les dimensions du contenant, comme dans un problème de découpe. On peut aussi rencontrer des cas où le contenant sert de support aux composants à positionner. C'est le cas par exemple dans le problème d'agencement des composants d'un camion, où ces composants doivent être placés sur le châssis [Yin00, MFG08]. Enfin, le problème peut présenter plusieurs contenants. Cette situation peut soit signifier qu'il faut identifier le contenant où placer chaque composant, soit qu'il faut trouver le nombre minimum de contenants pouvant accueillir les composants.

Les inconnues du problème sont les variables de positionnement, ainsi que certains attributs des composants. Comme on le verra par la suite, ces variables de positionnement peuvent être de différents types et dépendent de la modélisation choisie.

Chaque problème de placement présente au moins des contraintes de non-chevauchement entre composants et, dans la plupart des cas, des contraintes d'appartenance au contenant. Ces contraintes définissent les contraintes de placement du problème. Des contraintes additionnelles de positionnement peuvent être considérées dans les problèmes d'agencement.

Les objectifs d'un problème sont ses critères d'évaluation. Ils sont exprimés comme des quantités numériques utilisées pour classer les différentes solutions proposées. Les objectifs sont des fonctions mathématiques explicites ou des résultats numériques définis par le concepteur. C'est le cas lorsque les critères ne peuvent être modélisés explicitement par des fonctions mathématiques. Par exemple, les critères esthétiques qui font intervenir le point de vue subjectif du concepteur ou d'un expert, peuvent rarement être modélisés par une fonction mathématique. Dans la suite de ce travail, l'évaluation de chaque objectif est supposée renvoyer une valeur numérique.

1.2.2 Définition des problèmes de placement

Les définitions des différents éléments d'un problème de placement permettent de définir ce dernier :

Étant donné un ensemble de composants ainsi qu'un ensemble de contenants, un problème de placement consiste à trouver l'ensemble des variables de positionnement des composants afin de minimiser un ensemble d'objectifs, tout en respectant des contraintes. Dans tous les cas, les contraintes incluent les contraintes de placement.

Un placement ou une solution peut être vu comme une affectation de l'ensemble des variables de positionnement ainsi que des attributs. Une solution réalisable ou admissible est une affectation de toutes les variables, telle que l'ensemble des contraintes de placement soit satisfait. Une solution réa-

lisable sera dite efficace si aucune autre solution n'est meilleure. Résoudre un problème de placement, c'est proposer une ou plusieurs solutions efficaces.

1.2.3 Vocabulaire des problèmes de placement

Dans la littérature, plusieurs mots ou expressions sont utilisés pour désigner la même chose ou bien un même mot peut désigner différentes choses. Le mot *placement* a été utilisé plusieurs fois dans la littérature : Wäscher *et al.* [WHS07] utilisent ce terme pour définir une catégorie de problèmes, pour laquelle une série de composants faiblement hétérogène doit être affectée à un ensemble fini de contenants. Dans ce travail ainsi que dans de nombreux travaux, le mot *placement* fait référence à une solution.

Les différents éléments d'un problème de placement ont été utilisés dans différents contextes dans la littérature. Par exemple, Zhang *et al.* [ZTS08] utilisent le terme *objet* pour désigner un composant à placer à l'intérieur d'un satellite, alors que les *objets* font référence au contenant dans la typologie C&P de Dyckhoff [Dyc90]. Dans les problèmes de *bin packing*, les termes objets, items ou encore pièces sont utilisés pour désigner les composants. Dans les problèmes d'intégration à grande échelle, le terme module désigne un composant rectangulaire [MFNK96].

1.3 Caractéristiques des problèmes de placement

Cette section présente les différents aspects de modélisation des problèmes de placement.

1.3.1 Techniques de placement

Une fois que les problèmes de placement ont été définis, les techniques de placement doivent être mise en place pour positionner l'ensemble des composants.

Pour comprendre les différentes techniques de placement qui ont été développées, intéressons-nous à un simple problème de placement unidimensionnel. Soient cinq palettes de dimensions identiques qui doivent être positionnées sur un axe de manière à optimiser un ou plusieurs objectifs. Une contrainte implicite du problème consiste à ce qu'aucun couple de palettes ne se chevauche. Pour caractériser cette contrainte, la notion de longueur de recouvrement est utilisée. Cette grandeur exprime la largeur de l'intervalle de recouvrement entre deux composants unidimensionnels. La figure 1.3 présente les cinq palettes à positionner, ainsi que la fonction pénalité pour la palette *e*. Cette fonction, linéaire par morceaux, est loin d'être une simple fonction convexe. La figure 1.4 illustre l'allure de la fonction contrainte pour les deux palettes *a* et *b* libres de se traduire. Les nombreux minima locaux de cette fonction montrent combien il peut être difficile pour un optimiseur de trouver une solution réalisable. Cette caractéristique récurrente des problèmes de placement a conduit au développement de méthodes de placement permettant de satisfaire les contraintes de non-chevauchements.

Deux techniques de placement ont été élaborées : les méthodes de placement légal et relaxé. Les méthodes de placement relaxé autorisent le non-respect des contraintes de placement lors de la construction de la solution. Les méthodes légales garantissent le non-chevauchement des composants, en utilisant des techniques de placement pour générer les solutions. Le terme *relaxé* fait référence aux contraintes de placement qui sont relaxées lors de la résolution des problèmes. Si une telle méthode est utilisée, les contraintes de placement peuvent ne pas être satisfaites à la fin de l'optimisation. Le terme *légal* est à mettre en opposition avec le terme *relaxé*. Le choix de la méthode de placement n'est pas dicté par la géométrie des composants, mais plutôt par le type d'objectifs et de contraintes du problème, comme nous le verrons par la suite.

1.3.2 Représentation des variables

Les variables de positionnement sont les inconnues du problème de placement. Ces variables peuvent être de différents types : continues, discrètes, binaires voire même des permutations. Il existe plusieurs méthodes pour représenter la position et l'orientation des composants dans un problème de placement. Ce choix dépend de plusieurs critères, comme la géométrie des composants, des objectifs

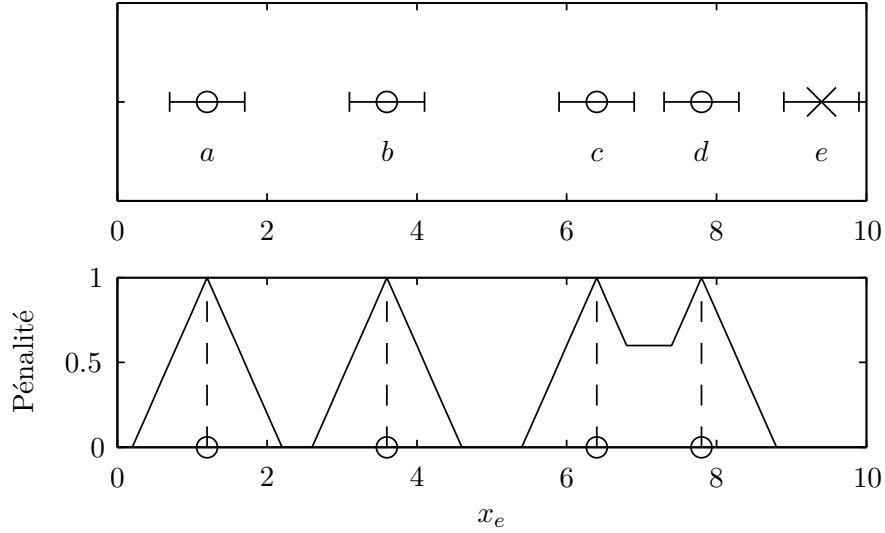
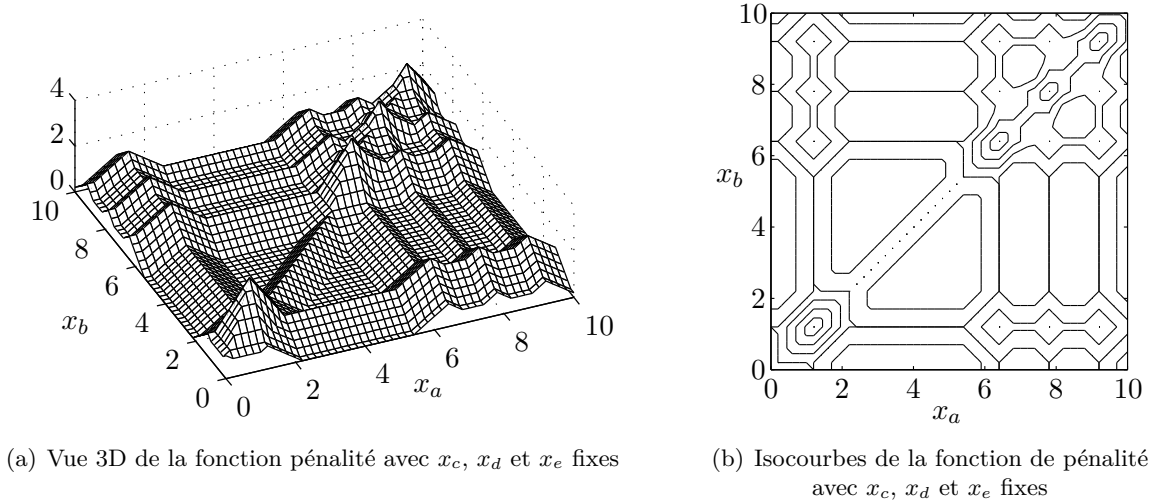


Figure 1.3 – Exemple d’une fonction de pénalité pour un problème de chargement de palettes de longueur unitaire. Quatre palettes (a, b, c, d) sont fixées et la cinquième est libre de se déplacer le long de l’axe. La fonction pénalité est calculée à partir des profondeurs de pénétrations entre composants. Cet exemple met en évidence le caractère multi-modal de la contrainte de non-chevauchement entre composants.



(a) Vue 3D de la fonction pénalité avec x_c , x_d et x_e fixes

(b) Isocourbes de la fonction de pénalité avec x_c , x_d et x_e fixes

Figure 1.4 – Représentation de la fonction contrainte pour le problème de chargement de palettes 1D avec 5 palettes dont 3 sont fixes : $x_c = 6.4$, $x_d = 7.8$ et $x_e = 9.2$.

et contraintes du problème. Ce choix détermine la topologie de l’espace de recherche, et conditionne les algorithmes d’optimisation qui pourront être utilisés par la suite.

Étant donné que l’allure de la fonction contrainte présente de nombreux minima locaux, une représentation des variables permettant de satisfaire les contraintes de placement est d’un grand intérêt. Les schémas d’encodage permettent de construire des solutions réalisables. Basés sur des règles de placement, les schémas d’encodage présentent l’avantage de satisfaire les contraintes de placement du problème. Ces schémas peuvent être utilisés lorsque les composants doivent être placés en contact les uns avec les autres ou bien lorsqu’ils doivent être placés à une certaine distance les uns des autres [GWF96].

1.3.2.1 Représentations basées sur l’utilisation de permutations

Une permutation est une liste ordonnée d’entiers tous différents, qui fait référence aux identifiants

des composants. L'information est représentée par la position de chaque indice dans la permutation. Le problème devient un problème combinatoire, où le nombre de solutions est fini mais très grand. Si m est le nombre de composants à positionner, alors une simple permutation peut générer $m!$ solutions.

Généralement, les permutations sont utilisées pour représenter l'ordre d'introduction des composants dans un contenant, comme pour les heuristiques *Bottom-Left* [Jak96], qui positionnent de manière séquentielle les composants le plus en bas à gauche. Principalement utilisée pour la résolution des problèmes C&P, cette heuristique est couplée à un algorithme d'optimisation combinatoire pour proposer des solutions.

1.3.2.2 Représentations basées sur l'utilisation de variables discrètes

Les variables discrètes sont utilisées dans différents contextes. Elles peuvent être utilisées pour positionner les composants sur une grille. Ce genre de représentation est utilisé lorsque les composants sont représentés sous forme de rectangles/parallélépipèdes, ou définis comme des ensembles de carrés/cubes adjacents. Cette représentation permet de vérifier simplement les contraintes de placement. Les variables binaires et discrètes peuvent aussi modéliser l'ensemble des différentes orientations possibles d'un composant. Dans beaucoup de problèmes, l'orientation des composants n'est pas une variable du problème et permet ainsi de réduire le nombre d'inconnues du problème. Les variables binaires sont aussi utilisées pour représenter la présence ou non d'un composant à l'intérieur d'un contenant. Les problèmes C&P utilisent ce genre de variables pour modéliser la sélection ou non d'un composant dans un contenant.

1.3.2.3 Représentations basées sur l'utilisation de variables réelles

L'utilisation de variables réelles permet de placer de manière continue les composants. Cela correspond à la représentation la plus naturelle des variables qui peut être utilisée pour tout type de problème. Toutefois, elle n'est pas particulièrement adaptée aux problèmes présentant certaines spécificités.

Le positionnement des composants à l'aide de variables réelles peut être absolu ou relatif. Le positionnement absolu d'un composant est utilisé lorsque sa position ne dépend d'aucun autre composant. En revanche, le positionnement relatif est utilisé lorsqu'un composant est fonctionnellement relié à un autre composant. Le positionnement relatif est employé dans les problèmes d'agencement avec certaines contraintes topologiques. Il permet de satisfaire facilement certaines contraintes de positionnement relatif, comme l'alignement de deux composants. Ce genre de contraintes pourrait être difficile à satisfaire s'il n'était pris en compte dans la modélisation.

1.3.3 Les objectifs des problèmes de placement

Les objectifs et les contraintes sont les critères utilisés pour qualifier les solutions. L'expression mathématique des objectifs et contraintes est cruciale. Les objectifs peuvent être de trois types : individuels (*p. ex.* placer ce composant le plus à droite), globaux (*c-à-d* s'appliquant sur l'ensemble des composants) et les objectifs d'interaction. Un objectif d'interaction fait intervenir une relation entre différents composants. On retrouve ce type d'objectifs dans les problèmes d'agencement, plus particulièrement dans les problèmes d'agencement de locaux ou de placement de composants dans une puce électronique.

Pour tous les problèmes C&P, l'objectif ou les objectifs sont globaux et s'exprimeront sur l'ensemble des composants. Une des caractéristiques des problèmes C&P porte sur ce caractère global des objectifs.

1.3.4 Les contraintes des problèmes de placement

Pour tous les problèmes d'optimisation, les contraintes permettent d'identifier les solutions réalisables et irréalisables. Pour les problèmes de placement, trois types de contraintes peuvent être identifiées :

- celles qui consistent à bloquer certains degrés de liberté des composants ;

- celles qui s’expriment comme une combinaison de plusieurs paramètres d’optimisation ;
- celles qui sont une combinaison d’objectifs.

Naturellement, les premières sont les plus simples à satisfaire. De plus, ces restrictions réduisent le nombre de degrés de libertés et par conséquent l’espace de recherche. Les secondes peuvent être vérifiées avant d’évaluer les objectifs. Enfin, les contraintes du troisième type nécessitent d’évaluer les objectifs pour savoir si les contraintes sont satisfaites.

Les contraintes de non-chevauchement entre composants ainsi que les contraintes d’appartenance des composants au contenant définissent les contraintes communes à tous les problèmes de placement. Quelques autres contraintes classiques que l’on peut rencontrer dans les problèmes d’agencement sont listées ci-après :

- contraintes de position ou d’orientation ;
- contraintes de proximité ;
- contraintes de restriction de domaine.

Plusieurs contraintes qui expriment des relations explicites entre les composants et le contenant doivent être prises en compte lors de la modélisation du problème. Cela permet de garantir leurs respects.

1.3.5 Évaluation des contraintes de non-chevauchement et d’appartenance

Dans les problèmes de placement 2D et 3D, les calculs de chevauchement et d’appartenance peuvent être très coûteux et représenter la majorité du temps de calcul. Pour un problème avec m composants, $(m^2/2 + m/2)$ tests sont nécessaires pour détecter le chevauchement et l’appartenance.

L’idéal consiste à utiliser une méthode de placement légal, où les contraintes de non-chevauchement sont automatiquement satisfaites. Elle doit tout de même s’assurer que tous les composants ont pu être placés. Dans le cas où une telle méthode ne peut être utilisée, il faut s’assurer que chaque composant est bien à l’intérieur du contenant et qu’aucune paire de composants ne se chevauche. Ces tests peuvent être effectués de différentes manières en fonction de la représentation géométrique choisie. Les problèmes avec des géométries régulières (rectangles, cercles, sphères,...) peuvent utiliser des formulations analytiques pour évaluer les contraintes. En revanche, lorsque la géométrie des composants devient complexe, différentes stratégies ont été développées : en 2D, des outils géométriques comme les polygones de non-recouvrement peuvent être utilisés pour détecter le chevauchement entre composants à moindre coût. En 3D, les géométries sont généralement approchées pour permettre une détection rapide des collisions. Parmi toutes les approximations géométriques possibles, les représentations sous formes de *voxels* (contraction anglophone de *volumetric pixels*), d’*octree* (contraction anglophone de *octant tree*) permettent de détecter facilement les collisions entre composants. Les différentes méthodes pour représenter la géométrie des composants sont détaillées dans le chapitre 4, présentant la méthode de placement proposée.

1.4 Les problèmes de découpe et de conditionnement

Les problèmes de découpe et de conditionnement (*Cutting and Packing problems*), forment une famille de problèmes à part entière. Dans la littérature française, l’expression *découpe et empaquetage* est aussi employée. Ils sont abrégés par la suite C&P.

Ce sont des problèmes dont les formulations verbale et mathématique sont simples, mais pour lesquels la combinatoire rend ces problèmes difficiles à résoudre. Les premiers travaux remontent à plus de quarante ans. Depuis, plusieurs classifications de ces problèmes ont été entreprises [Dyc90, WHS07].

1.4.1 Définition des problèmes de découpe et de conditionnement

Les problèmes de découpe et de conditionnement sont des problèmes de placement, pour lesquels les composants sont uniquement géométriquement reliés entre eux. En d’autres termes, il n’y a aucune interaction explicite entre les composants. Tous les composants présentent les mêmes types d’attributs. Objectifs et contraintes peuvent toujours être modélisés comme des fonctions mathématiques et sont définis globalement, c’est-à-dire qu’ils impliquent l’ensemble des attributs des composants.

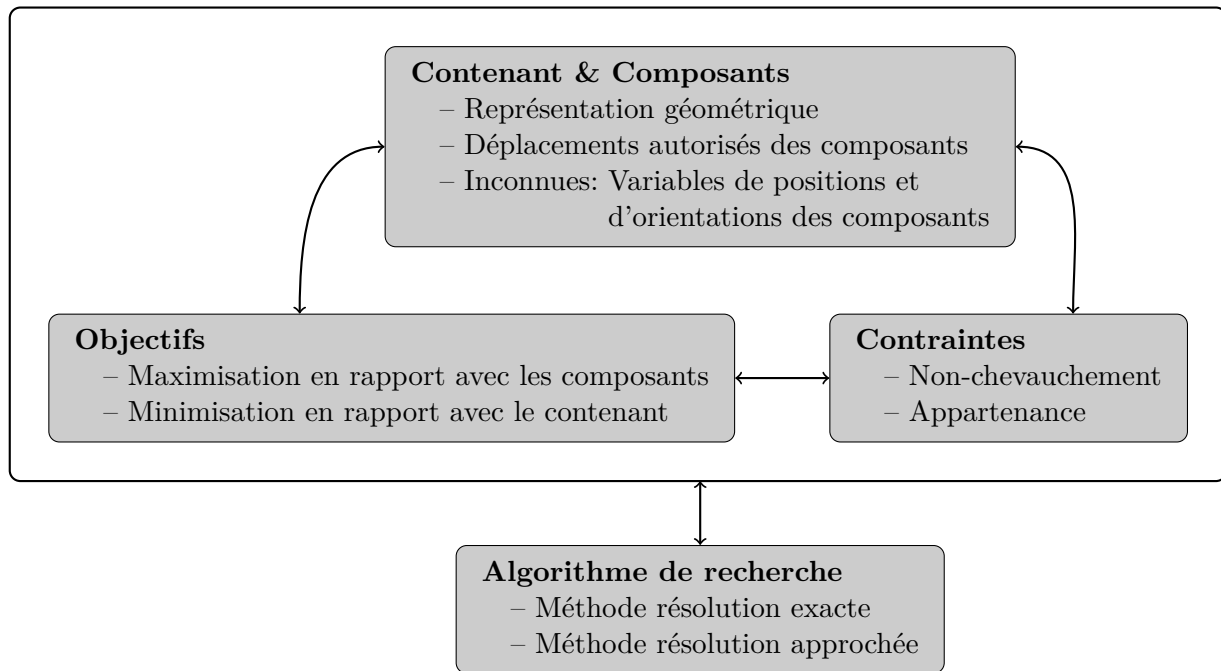


Figure 1.5 – Constituants principaux d'un problème de découpe et de conditionnement.

La figure 1.5 présente les différents éléments d'un problème C&P. L'idée derrière chaque problème C&P est la compacité : soit le nombre de composants à placer dans le contenant doit être maximisé, soit le nombre de contenants ou les dimensions de ceux-ci doivent être minimisés. Par conséquent, chaque schéma de résolution essaie de tirer parti de cette caractéristique récurrente. C'est la différence majeure avec les problèmes d'agencement, pour lesquels les objectifs n'expriment pas forcément un objectif de compacité.

Les problèmes C&P englobent une large variété de problèmes qui ont tous été étudiés depuis plusieurs années. En 2007, Wäscher *et al.* [WHS07] ont identifié plus de 400 articles traitant de ces problèmes. Ces problèmes sont bien connus et plusieurs typologies ont été développées pour classer ces problèmes, rendant leurs identifications plus faciles.

Les problèmes C&P peuvent être décomposés en deux types de problèmes, les problèmes de décision et les problèmes de minimisation / maximisation. Un problème de décision est un problème pour lequel la réponse est binaire : *oui* ou *non*, par exemple, *est-il possible de placer l'ensemble de ces composants à l'intérieur de ce contenant ?*. Ces problèmes sont des problèmes \mathcal{NP} -complets [GJ79]. \mathcal{NP} est l'abréviation de *Non-déterministe polynomial*. Cela signifie que les solutions associées au problème de décision correspondant peuvent être vérifiées à l'aide d'un algorithme polynomial. Le second type de problème implique une réponse numérique : *combien de ces composants peut-on placer à l'intérieur de ce contenant ?*. Les problèmes de décision sont généralement utilisés pour résoudre les problèmes C&P de minimisation / maximisation. Un exemple classique de cette situation se retrouve dans les problèmes de découpe de formes irrégulières, où l'on résout successivement une suite de problèmes de décision. La plupart de ces problèmes sont \mathcal{NP} -difficiles [GJ79].

Il peut être aussi noté que la plupart des problèmes C&P sont mono-objectifs, spécificité qui a été exploitée pour construire les différentes typologies. Les problèmes multi-objectifs de cette catégorie impliquent plusieurs fois le même type d'objectifs, comme par exemple pour le problème de sac à dos multi-objectif. Par conséquent, les problèmes C&P multi-objectifs peuvent être rangés dans les différentes typologies développées pour les problèmes mono-objectifs.

Les problèmes C&P représentent une branche des problèmes de placement, pour lesquels une formulation est facile à obtenir, mais nécessitent des méthodes de résolution efficaces pour obtenir des solutions optimales ou quasi-optimales. Les sections suivantes présentent les différentes typologies de la littérature proposées ainsi que les différents problèmes C&P.

1.4.2 Typologies des problèmes de découpe et de conditionnement

Dyckhoff [Dyc90] a été le premier à proposer une typologie des problèmes de découpe et de conditionnement. Cette typologie est basée sur quatre paramètres qui permettent d'identifier 96 problèmes. (Voir TAB. 1.1). Le premier paramètre sert à identifier la dimension du problème. Le second renseigne sur le type d'affectation, à savoir si tous les composants doivent être placés à l'intérieur du ou des contenants ou si seulement une sélection de ces composants doit être positionnée. Dans le premier cas, le problème est une minimisation effectuée sur les contenants (taille ou nombre). Dans le second cas, il s'agit d'une maximisation effectuée sur les composants à positionner. Concrètement, il s'agira de maximiser l'espace d'utilisation ou une fonction profit en sélectionnant un sous-ensemble des composants. Le troisième paramètre fournit des indications sur le type de contenant. Enfin le quatrième paramètre contient des informations sur la nature des composants, à savoir s'ils sont identiques, faiblement différents ou fortement différents. La redondance de certains problèmes, le manque de reconnaissance internationale ont poussé Wäscher *et al.* [WHS07] à améliorer cette typologie. Entre temps, Dyckhoff et Finke [DF92] ainsi que Dowsland *et al.* [DD92] ont proposé un tour d'horizon des problèmes de découpe et de conditionnement. Dyckhoff *et al.* ont proposé une bibliographie annotée en 1997 [DST97].

1. Dimension du problème
 - 1 Problème unidimensionnel
 - 2 Problème bidimensionnel
 - 3 Problème tridimensionnel
 - N Problème multi-dimensionnel avec $N > 3$
2. Type d'affectation
 - B Tous les contenants et une sélection des composants
 - V Une sélection des contenants et tous les composants
3. Nature des contenants
 - O Un seul contenant
 - I Plusieurs contenants identiques
 - D Plusieurs contenants différents
4. Nature des composants
 - F Quelques composants
 - M De nombreux composants très différents entre eux
 - R De nombreux composants assez similaires entre eux
 - C Composants identiques

Table 1.1 – Classification des problèmes de découpe et de conditionnement proposée par Dyckhoff [Dyc90].

La figure 1.6 présente le découpage des problèmes C&P effectué par Dyckhoff [Dyc90]. Il montre que ces problèmes touchent aussi bien les domaines de l'industrie de la découpe que les systèmes de production. La figure 1.7 présente la classification de Wäscher *et al.* des problèmes C&P classiques. Cette classification est orientée sur l'objectif du problème, à savoir une maximisation sur les composants ou une minimisation à effectuer sur le contenant. Les deuxième et troisième critères concernent la géométrie du contenant et des composants. À partir de ces trois critères, plusieurs problèmes types sont établis. Chacun de ces problèmes peut être raffiné en fonction des spécificités de chaque problème.

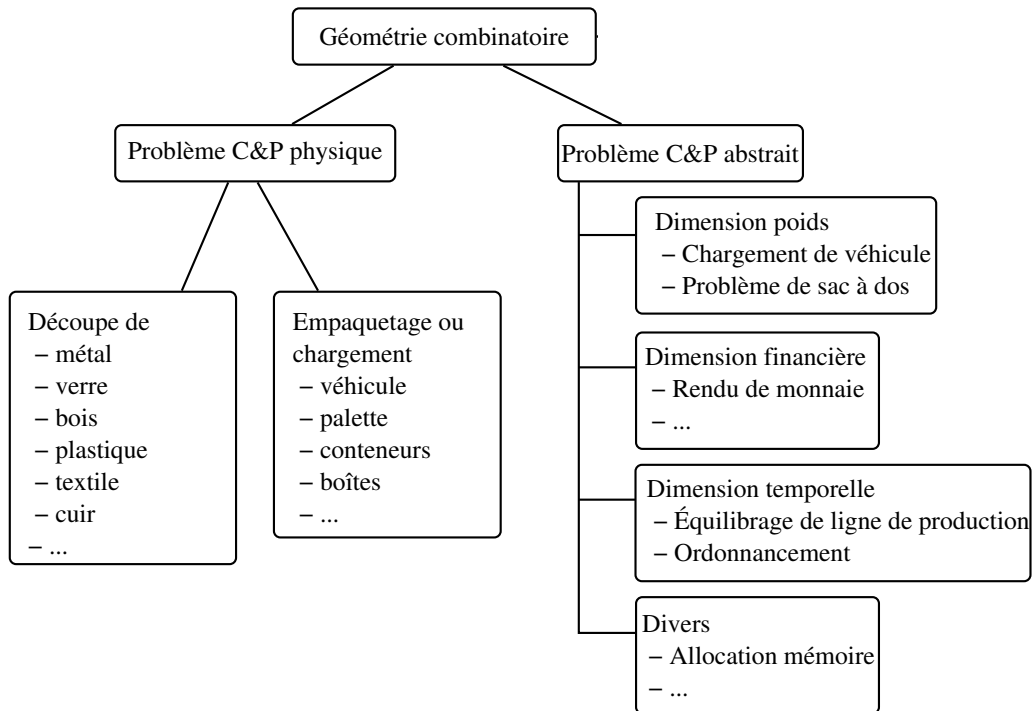


Figure 1.6 – Phénoménologie des problèmes de découpe et de conditionnement proposée par Dyckhoff [Dyc90].

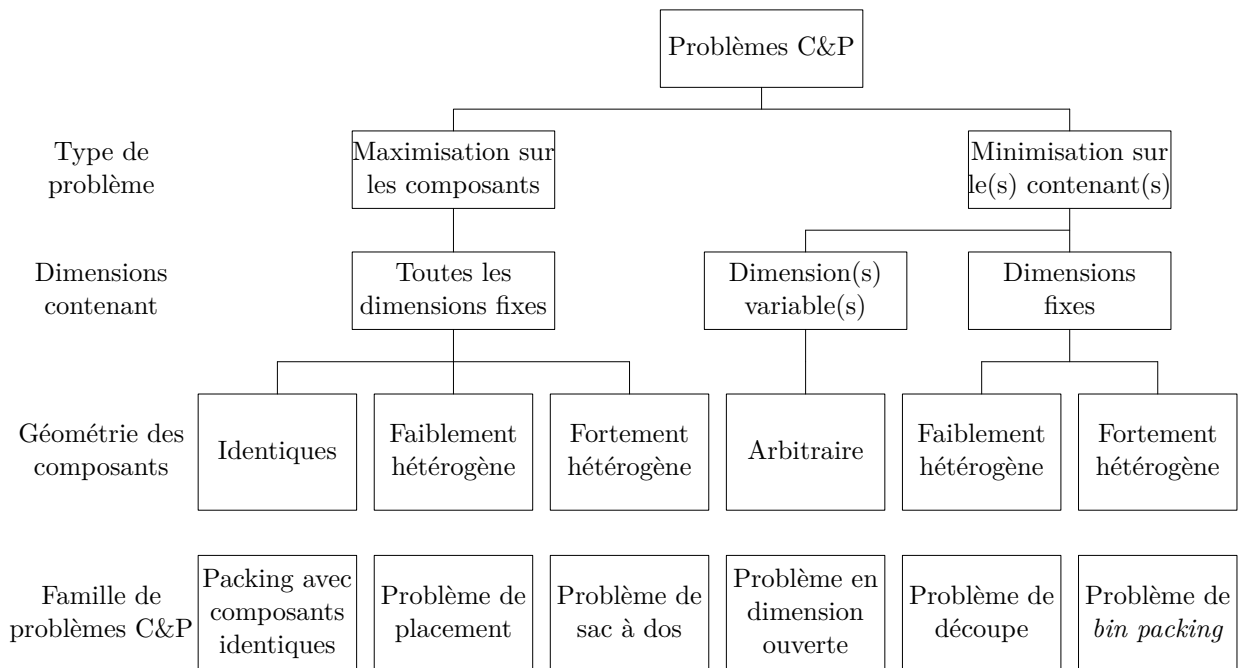


Figure 1.7 – Classification des problèmes de découpe et de conditionnement selon Wäscher *et al.* [WHS07].

1.4.3 Problèmes de découpe

Les problèmes de découpe trouvent de nombreuses applications dans l'industrie, et présentent un intérêt économique certain.

1.4.3.1 Problème de découpe unidimensionnelle

Classiquement, le problème de découpe unidimensionnelle (*1D cutting stock problem*) consiste à trouver le nombre minimal de contenants de dimensions identiques pouvant contenir un ensemble donné de composants. Les contenants peuvent être vus comme des stocks dans lesquels différents éléments (composants) doivent être découpés. La différence entre le problème de *bin packing* et de découpe réside dans la variété des composants à placer ou à découper. Pour le problème de *bin packing*, les composants ont des longueurs variées, alors que pour le problème de découpe, les composants sont relativement identiques. Cette différence entraîne des modifications de la formulation et de la résolution du problème. Le problème est formulé comme un problème de programmation linéaire en nombres entiers et est généralement résolu avec une méthode de génération de colonnes [GG63].

1.4.3.2 Problème de découpe bidimensionnelle

Ce problème constitue la généralisation directe du problème précédent en 2D, où un ensemble de rectangles doit être découpé dans un contenant de largeur fixe et de hauteur minimale. La résolution de ces problèmes est aussi bien basée sur des heuristiques [BKW04] que sur des méthodes exactes [CMWX08].

1.4.4 Problèmes de sac à dos

1.4.4.1 Problème de sac à dos unidimensionnel

Le problème du sac à dos 1D est un problème de recherche opérationnelle très répandu et étudié depuis de nombreuses années [KPP04]. Le problème consiste à sélectionner un ensemble de composants ayant chacun un poids et un profit de manière à maximiser le profit total tout en respectant une contrainte de poids maximum. À chaque composant est associé une variable binaire qui modélise la sélection ou non du composant dans le sac à dos. Les méthodes de résolution exacte pour ce problème sont la programmation dynamique et la procédure d'évaluation et de séparation. Les méthodes de résolution approchée regroupent les algorithmes génétiques, les algorithmes de colonies de fourmis, et bien d'autres méthodes encore. Le problème possède de nombreuses variantes, qui font toutes l'objet de plusieurs recherches.

1.4.4.2 Problème de sac à dos multi-dimensionnel

Le problème de sac à dos peut être décliné en 2D et en 3D. En 2D, les composants et le contenant sont des rectangles alignés sur le système d'axe. En 3D, les éléments sont des parallélépipèdes rectangles. Dans le cas général, on parle de boîtes. Le problème de chargement de containers peut être vu comme un problème de sac à dos multi-dimensionnel. Il existe deux types de problèmes de chargement de containers :

- les chargements de type sac à dos (*Knapsack Loading*) : Chaque boîte a un profit associé, et le problème consiste à choisir un ensemble de boîtes qui maximise le profit. Si le profit d'une boîte est égal à son volume, alors le problème correspond à une minimisation de l'espace perdu, c'est le problème classique du chargement de containers. Des heuristiques de résolution de ce problème ont été proposées par Gehring *et al.* [GMM90], Scheithauer [Sch92] et plus récemment par Egeblad *et al.* [EP09] ;
- les problèmes minimisant les dimensions du contenant : Toutes les boîtes doivent être placées à l'intérieur d'un seul et même container, dont deux des dimensions sont fixées (hauteur H et profondeur P) et la longueur L étant variable. L'objectif est de trouver une solution qui permette de placer l'ensemble des boîtes à l'intérieur du container tout en minimisant sa longueur.

Plusieurs algorithmes ont été comparés par Bischoff et Marriott [BM90]. Ce problème est comparable au problème de découpe 2D, ou encore au problème d'assortiment qui consiste à trouver les dimensions minimales d'un contenant devant accueillir une série de composants identiques.

Ces problèmes sont respectivement identifiés par $D/B/O/M$ et $D/V/O/M$ selon la classification de Dyckhoff [Dyc90]. D indique ici la dimension du problème ($D \in \{1, 2, 3\}$). Le problème de sac à dos à deux dimensions est abrégé $2DKP$ pour *2D Knapsack Problem* et le problème tridimensionnel $3DKP$. La figure 1.8 présente les solutions obtenues lors de la résolution de deux problèmes de chargement de containers de type sac à dos.

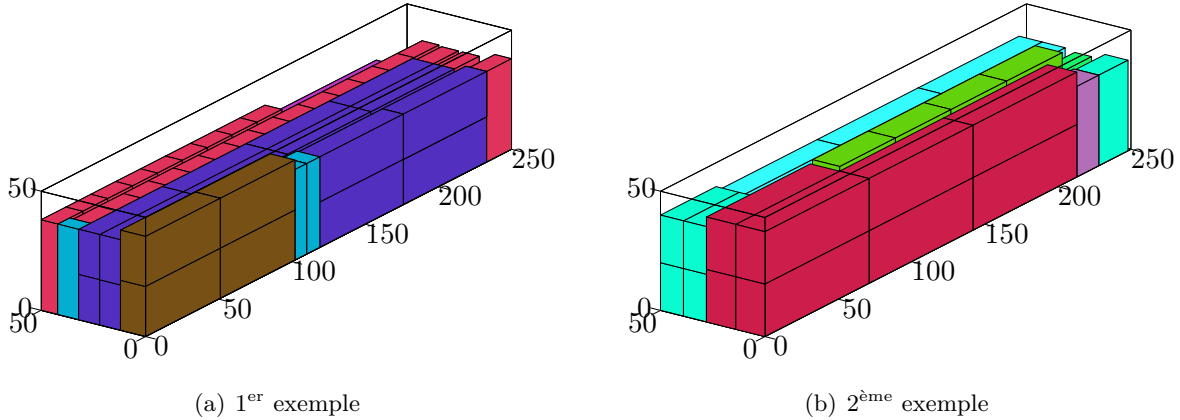


Figure 1.8 – Solutions de deux problèmes de chargement de containers.

1.4.5 Le problème de *bin packing* à deux dimensions

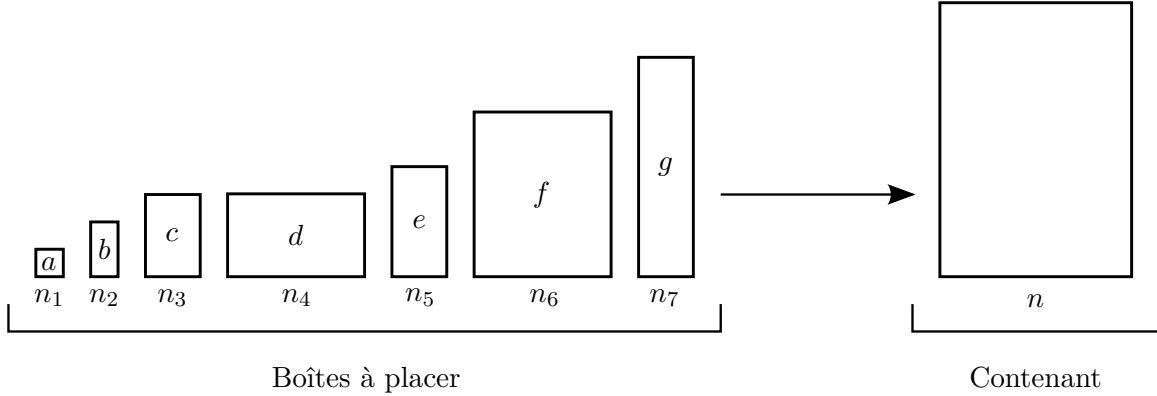
Le problème de *bin packing* ou empaquetage 2D ($2BP$) consiste à trouver le nombre minimum de contenants rectangulaires identiques pouvant accueillir une série de composants rectangulaires. Ces derniers sont rangés de telle façon que leurs arêtes soient parallèles à celles des contenants (on parle de rangement orthogonal). Le problème constitue une extension directe du problème de *bin packing* 1D. La figure 1.9 présente un exemple avec une série de sept types de composants à placer dans un nombre minimum de contenants identiques. Ce problème a de nombreuses applications industrielles, en particulier dans l'industrie de la découpe et dans la logistique.

Lorsque l'ensemble des composants a une hauteur identique, on retrouve le problème de *bin packing* à une dimension ($1BP$). Puisque $1BP$ est connu comme étant \mathcal{NP} -difficile, il en va naturellement de même pour $2BP$. Lodi *et al.* [LMV99, LMV02] ont proposé la typologie suivante permettant de catégoriser les différents problèmes de *bin packing*. Cette typologie est basée sur la prise en compte des rotations orthogonales des composants ainsi que sur le type de découpe. La rotation de 90° des rectangles permet d'augmenter la taille de l'espace de recherche, ainsi que la probabilité de trouver une meilleure solution. Enfin dans les problèmes de découpe, une contrainte de type guillotine peut être imposée : l'ensemble des composants doit être obtenu à partir d'une séquence de coupes horizontales et verticales du contenant [PM09]. Le codage de Lodi *et al.* se résume de la manière suivante :

- **2BP|OG** : les composants sont orientés (**O**), le découpage *guillotine* (**G**) est requis ;
- **2BP|RG** : les composants peuvent subir une rotation de 90° (**R**) et le découpage *guillotine* (**G**) est requis ;
- **2BP|OF** : les composants sont orientés (**O**) et le découpage est *libre* (**F**) ;
- **2BP|RF** : les composants peuvent subir une rotation de 90° (**R**) et le découpage est *libre* (**F**) ;

Le problème de 2D-*bin packing* a fait l'objet de nombreuses études et de nombreuses heuristiques ont rapidement été proposées : Chung *et al.* [CGJ83], Berkey et Wang [BW87], Frenk et Galambos [FG87] ont présenté des algorithmes gloutons en une et deux phases qui permettent d'obtenir des solutions approchées très rapidement. Bengtsson [Ben82] et El-Bouri *et al.* [EBPBA94] ont pris en compte une rotation éventuelle de 90° des composants. Pour une résolution exacte, on utilise essentiellement la programmation linéaire en nombres entiers pour résoudre ce problème. Lorsque l'instance traitée

est de faible taille, la formulation de Kantorovich [Kan60] peut être utilisée. Lorsque le nombre de composants est grand, on utilise plutôt une résolution par génération de colonnes utilisant le modèle de Gilmore et Gomory [GG61, GG63].



Exemple de solution pour $n_1 = 3$, $n_2 = 3$, $n_3 = 2$, $n_4 = 2$, $n_5 = 2$, $n_6 = 1$, $n_7 = 1$ et $n = 2$

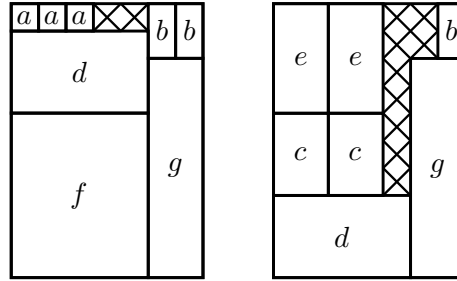


Figure 1.9 – Exemple de problème d’empaquetage (2DBP)

1.4.6 Problème de chargement de palettes

Le problème de chargement de palettes bidimensionnel consiste à placer un ensemble de composants rectangulaires faiblement différents dans un nombre minimal de palettes rectangulaires identiques. Le problème est similaire au problème de *bin packing* 2D excepté que les composants sont identiques. Cette différence génère des modélisations différentes, qui font de ce problème un problème à part entière [LLM03, BMN05].

1.4.7 Problème de découpe de formes irrégulières

Le problème de découpe de formes irrégulières (*Nesting problem*) consiste à positionner un ensemble de composants 2D de géométries complexes dans un contenant ayant des dimensions minimales, l’objectif étant de minimiser les chutes générées par la découpe des composants. Le problème est principalement rencontré dans les industries textile et métallique. Dans le cas où le contenant est de forme rectangulaire, on parle de problème de découpe en bande (*Strip packing problem*). Le problème est un problème \mathcal{NP} -difficile [NO03] et peut être classé comme 2D-ODP (*two-dimensional irregular open dimension problem*) selon la typologie de Wäscher [WHS07]. Comme pour les autres problèmes, de nombreuses variantes existent. Certaines sont des cas particuliers, comme lorsque l’ensemble des composants est identique [CGO09]. D’autres sont des extensions nécessitant une généralisation des modélisations proposées, comme pour les problèmes de découpes périodiques [Nie07].

De nombreux algorithmes de résolution ont été développés. Ces algorithmes sont aussi bien des méthodes relaxées (2D Nest [ENO07], ILSQN [IYN09]) que légales (*Beam Search for nesting problems* [BS07]).

1.4.8 Problèmes de conditionnement 3D d'objets de géométries quelconques

Le problème de conditionnement 3D d'objets de géométries quelconques, aussi connu sous le nom de *3D packing*, consiste à placer un maximum de composants de formes irrégulières dans un contenant lui aussi de forme quelconque. Ce problème trouve des applications dans des problèmes de chargement de coffre ou encore dans le prototypage rapide par frittage LASER (*Selective LASER Sintering – SLS*). Le prototypage rapide est un terme originellement utilisé pour la production de prototypes de modèles CAO. Aujourd'hui, les technologies de prototypages rapides sont utilisées pour la fabrication de composants. La figure 1.10 illustre ce procédé de fabrication. Les objets sont fabriqués par addition successive de couches. Un rouleau vient étaler uniformément une couche de poudre. Les zones produites sont chauffées à l'aide d'un LASER, qui solidifie la poudre. Une fois la couche réalisée, un système de pistons permet de faire progresser la fabrication des composants. La poudre qui n'a pas été chauffée permet de maintenir les objets en cours de réalisation. Ce procédé peut durer plusieurs heures, c'est pourquoi un maximum d'objets doit être fabriqué en même temps.

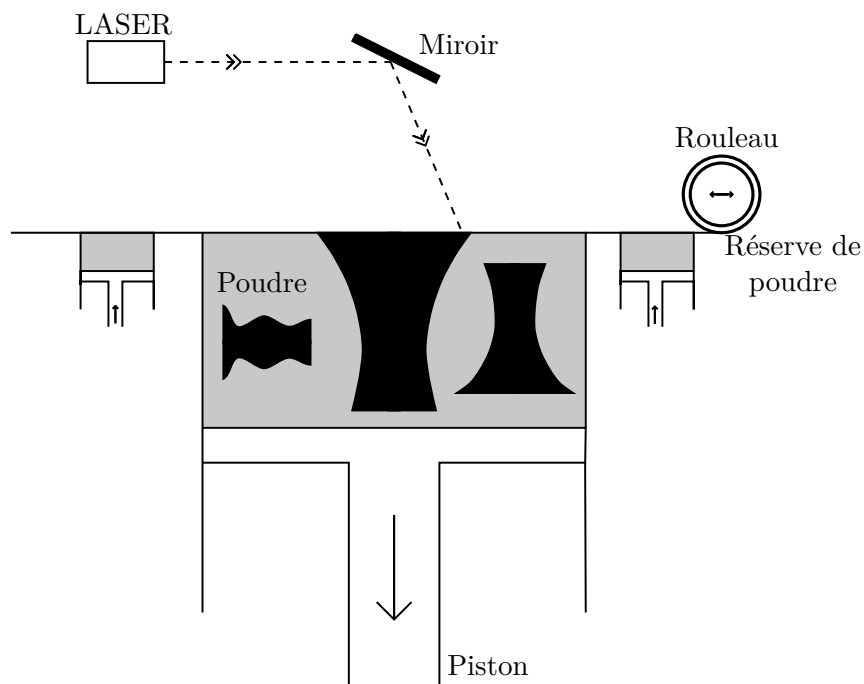


Figure 1.10 – Mise en forme par procédé de frittage.

La littérature sur ces problèmes est assez récente, les premiers articles datent d'une dizaine d'années. Quelques-unes des méthodes de résolution développées sont présentées ici. Les symboles *L* ou *R* de chaque méthode indiquent si la méthode de placement développée est *Légale* ou *Relaxée*.

- Ikonen *et al.* [IBK⁺97] ont utilisé un algorithme génétique et des boîtes englobantes pour représenter les objets. (L)
- Cagan *et al.* [CDY98] utilisent un algorithme de recuit simulé [KGV83] et une décomposition *octree* pour la représentation des objets. (R)
- Hur *et al.* [HCLC01] ont présenté une approche par *voxels*, où les variables d'optimisation représentent l'ordre d'introduction des composants. (L)
- Stoyan *et al.* [SGS⁺05] ont proposé un algorithme pour le placement de polyèdres (polyèdres convexes) à l'intérieur d'un parallélépipède, l'objectif étant de minimiser la hauteur nécessaire au placement de l'ensemble des composants. (L)
- Egeblad *et al.* [ENO07] ont adapté la technique de placement relaxée développée en 2D basée sur la métaheuristique *Guided Local Search*. (R)
- Tiwari *et al.* [TFF08] ont développé une méthode capable de gérer tout type de composants 3D avec des rotations libres. Leur méthode repose sur une décomposition des composants en *voxels*, une heuristique de placement appelée *BLBF*, pour *Bottom-Left-Back Fill* couplée à un

algorithme génétique. (L)

Cette dernière méthode semble actuellement la plus performante et mérite qu'on s'y attarde. L'heuristique *BLBF* permet de placer séquentiellement les composants. Le premier est initialement placé le plus en bas, à gauche au fond. Le composant suivant est aussi placé ensuite le plus en bas à gauche, et il est déplacé dans les trois directions de l'espace de manière méthodique, jusqu'à ce qu'une place soit trouvée. Le processus est répété jusqu'à ce que l'ensemble des composants soit placé, ou qu'il n'y ait plus de place. La figure 1.11 illustre la méthode de placement utilisée pour la résolution des problèmes de compactage sur un exemple 2D. Les différentes positions du composant en forme de + sont testées jusqu'à ce qu'il n'intersecte plus aucun carré représentant le contenant. En 3D, l'adaptation de la méthode permet de traiter des géométries vraiment complexes comme le montre la figure 1.12.

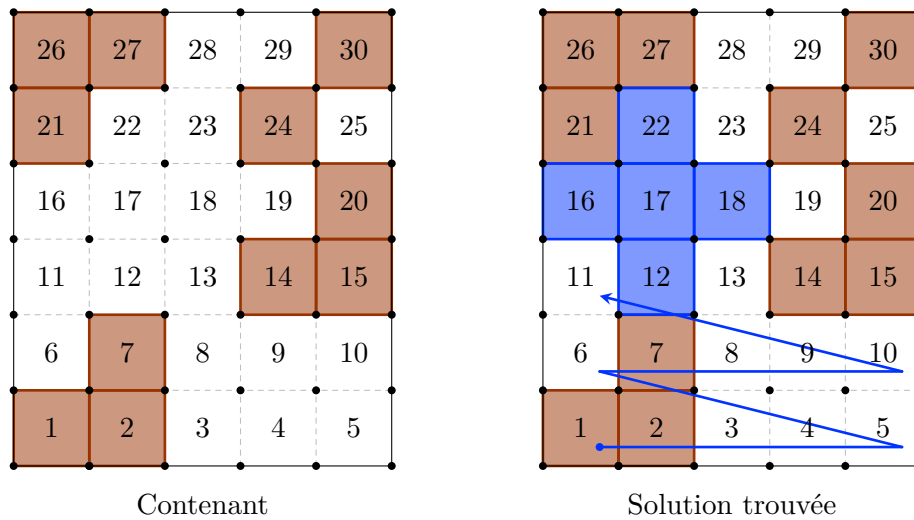


Figure 1.11 – Solution trouvée par l'heuristique de placement *Bottom-Left*. L'ensemble des voxels est parcouru jusqu'à ce qu'une solution réalisable soit trouvée.

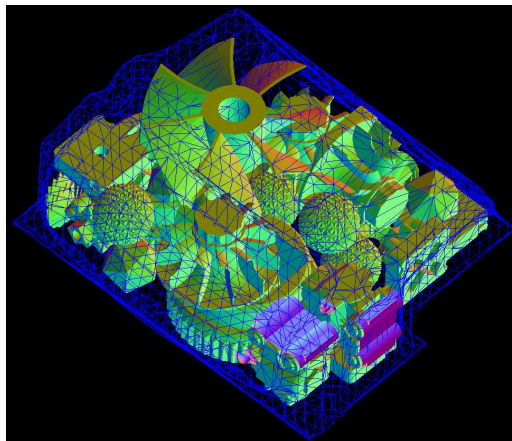


Figure 1.12 – Exemple de résolution de problème de packing 3D. La voxelisation du contenant et des composants permet de traiter de toute sorte de géométrie. Image empruntée à Tiwari *et al.* [TFF08].

1.5 Les problèmes d'agencement

Un problème d'agencement est défini comme un problème de placement, pour lequel un positionnement relatif des composants à l'intérieur du contenant est recherché. Dans de tels problèmes, les

composants et le contenant sont fonctionnellement et géométriquement reliés entre eux. Comme les problèmes C&P, les problèmes d'agencement sont \mathcal{NP} -difficiles.

La figure 1.13 présente les différents éléments d'un problème d'agencement. On retrouve sur cette figure les éléments des problèmes C&P, auxquels ont été ajoutés les spécificités liées aux problèmes d'agencement.

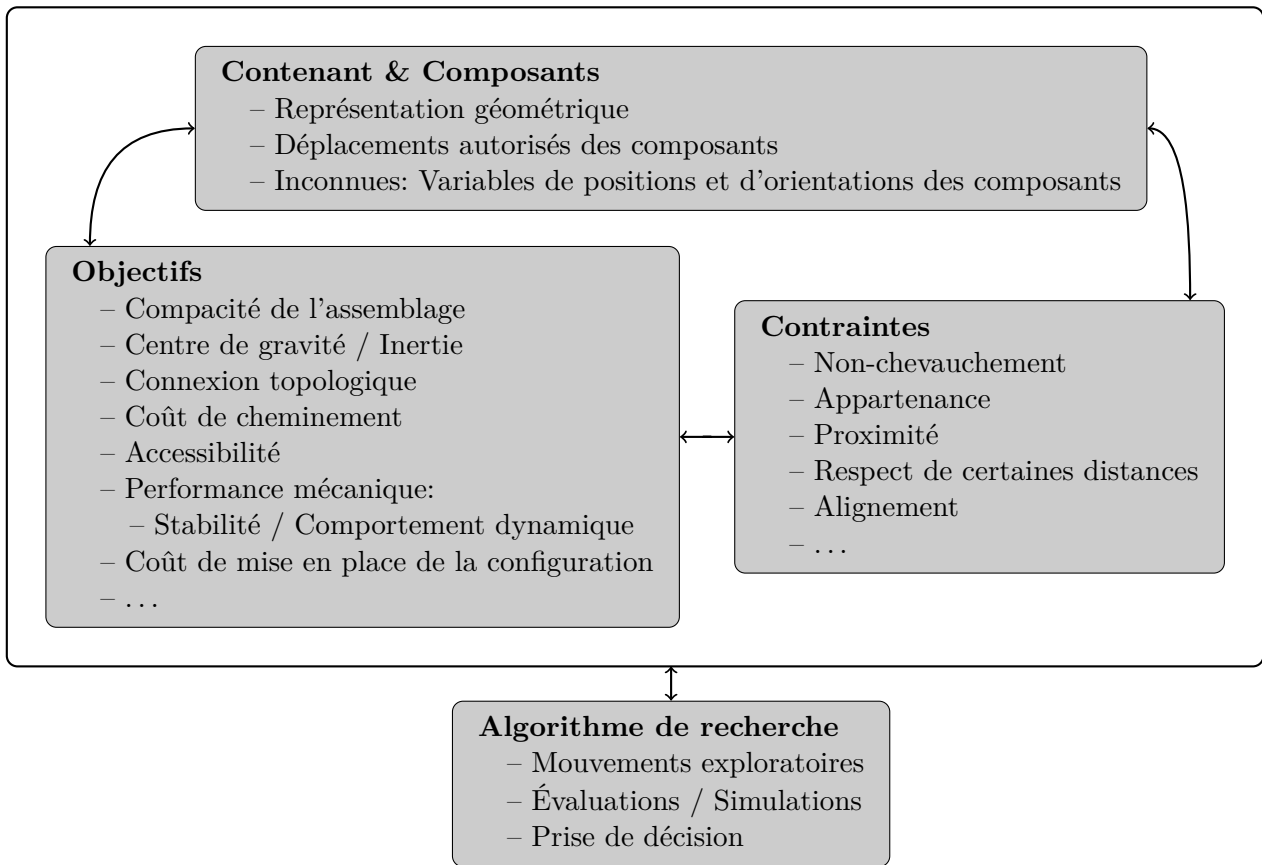


Figure 1.13 – Constituants principaux d'un problème d'agencement.

La plupart des techniques de résolution utilisées pour les problèmes C&P ne peut plus être utilisée ici. En revanche, les problèmes C&P peuvent être vus comme des cas particuliers des problèmes d'agencement. Par conséquent, les techniques de résolution développées pour la résolution d'agencement peuvent être utilisées pour la résolution des problèmes C&P.

Les problèmes d'agencement sont généralement présentés comme des problèmes multi-objectifs avec des objectifs contradictoires. Toutefois, la plupart des méthodes de résolutions proposées transforment les problèmes d'optimisation multi-objectif en une optimisation mono-objectif non contrainte. Un des objectifs de cette thèse sera la modélisation des problèmes multi-objectifs.

Les problèmes classiques d'agencement concernent l'agencement de locaux, l'intégration de puces à grande échelle, l'agencement 3D dans les problèmes d'ingénierie.

1.5.1 Problèmes d'agencement bidimensionnel

1.5.1.1 Problèmes d'agencement de locaux

Le problème d'agencement de locaux, ou d'agencement d'ateliers (*Facility layout problems*, abrégé *FLP*) consiste à agencer une série de composants 2D en optimisant les interactions entre ces composants. Ils sont traités comme des problèmes d'optimisation combinatoires.

Les composants rencontrés dans ce genre de problèmes sont principalement des rectangles alignés sur le système d'axe, ou des ensembles de carrés adjacents. Ces problèmes sont caractérisés par les interactions entre composants, principalement exprimées comme des minimisations de flux.

Le choix de la modélisation dépend ici de nombreux paramètres. Drira *et al.* [DPHG07] ont proposé une étude sur les différentes manières dont le problème de placement de locaux a été abordé. La figure 1.14 permet d'analyser la hiérarchie des problèmes d'agencement de locaux ou d'ateliers.

Les méthodes de résolution sont quasiment toutes basées sur des schémas d'encodage qui permettent de satisfaire les contraintes de non-chevauchement. Aiello *et al.* [AEG06] ont proposé un algorithme génétique multi-objectifs avec un codage de type permutation pour générer plusieurs solutions et laisser au concepteur le choix du meilleur compromis. La permutation fournit l'ordre d'introduction des pièces (composants) dans l'atelier (contenant). Des méthodes d'aide à la décision ont été introduites comme *AHP* (*Analytical Hierarchical Process*) pour choisir une solution parmi les meilleures. Wang *et al.* [WHK05] utilisent une méthode de remplissage de l'espace appelée *Space filling curve*. Basée sur le critère de respect d'aire des locaux, cette méthode de placement des locaux est présentée sur un exemple figure 1.15. Les locaux n'ont pas forcément une géométrie rectangulaire, ce qui peut être un avantage ou un inconvénient, comme nous le verrons dans le chapitre 3.

1.5.1.2 Problèmes d'intégration à grande échelle

Le problème d'intégration à grande échelle (*Very Large Scale Integration*, abrégé *VLSI*) fait l'objet de nombreuses études. Il consiste à placer un ensemble de composants rectangulaires de manière à minimiser une ou plusieurs fonctions objectifs. Les composants sont généralement appelés modules. Ces problèmes sont caractérisés par le très grand nombre de composants rencontrés, la forte interaction des composants ainsi que des fonctions objectifs très similaires d'un problème à l'autre. Classiquement, on cherche à minimiser la longueur des connexions entre l'ensemble des composants, pour minimiser les temps de communication. Depuis peu, ces problèmes sont traités de manière multi-objectif. Le premier objectif reste toujours la minimisation des longueurs de connexions, le second objectif peut être la minimisation de la chaleur dissipée par les composants. Dans les systèmes embarqués, on souhaite aussi maximiser la première fréquence de vibration des cartes électroniques.

Suivant la géométrie des composants, il existe plusieurs types de problèmes VLSI. La figure 1.16 présente les différents types de problèmes. Les cas (a) à (d) représentent des problèmes purement combinatoire. Pour les problèmes (a) et (b), les modules sont de taille identique. Le problème devient alors un problème d'affectation, problème purement combinatoire dans lequel la géométrie n'intervient plus. Dans le cas mono-objectif, l'algorithme hongrois (*Hungarian algorithm*) permet de résoudre le problème avec une complexité en $O(n^3)$ [Kuh55, Mun57]. Deb *et al.* [DJGM04] ont développé une version multi-objectif basée sur un algorithme génétique pour résoudre ce problème là. Pour les problèmes (c) et (d), une méthode de placement séquentiel positionnant les modules les uns à la suite des autres est utilisée. Ils sont modélisés sous forme combinatoire et résolus à l'aide d'heuristiques. Le cas général, où les modules sont de taille différente, est représenté sur la sous-figure (e). Ce cas peut être traité de manière combinatoire ou avec une formulation en variables réelles. La modélisation de ces problèmes est basée sur le constat suivant : derrière l'objectif de minimisation de la longueur des connexions se cache un objectif de compacité. Plus l'assemblage des composants sera compact, plus petite sera la somme des longueurs de connexions. À partir de ce constat, différents schémas d'encodage ont été développés. L'information de position des composants est alors représentée dans une chaîne codante, utilisée pour définir les coordonnées de chaque composant. La chaîne codante peut être une permutation, un ensemble de permutations ou encore une expression polonaise [OI98]. Les schémas d'encodage les plus connus sont *Inverse Polish Notation* (IPN), *Ordered-Tree* [GCY99] (*O-Tree*) et *Sequence Pair* [Pis07] (*SP*). Une liste plus exhaustive des différents schémas d'encodage est présentée en annexe table A.3. L'algorithme de recuit simulé [KGV83] est principalement utilisé pour optimiser l'agencement des circuits [HS87, Sec88, Ege03]. Un exemple d'agencement d'un ensemble de rectangles encodé à l'aide d'une notation polonaise inverse est présenté figure 1.17.

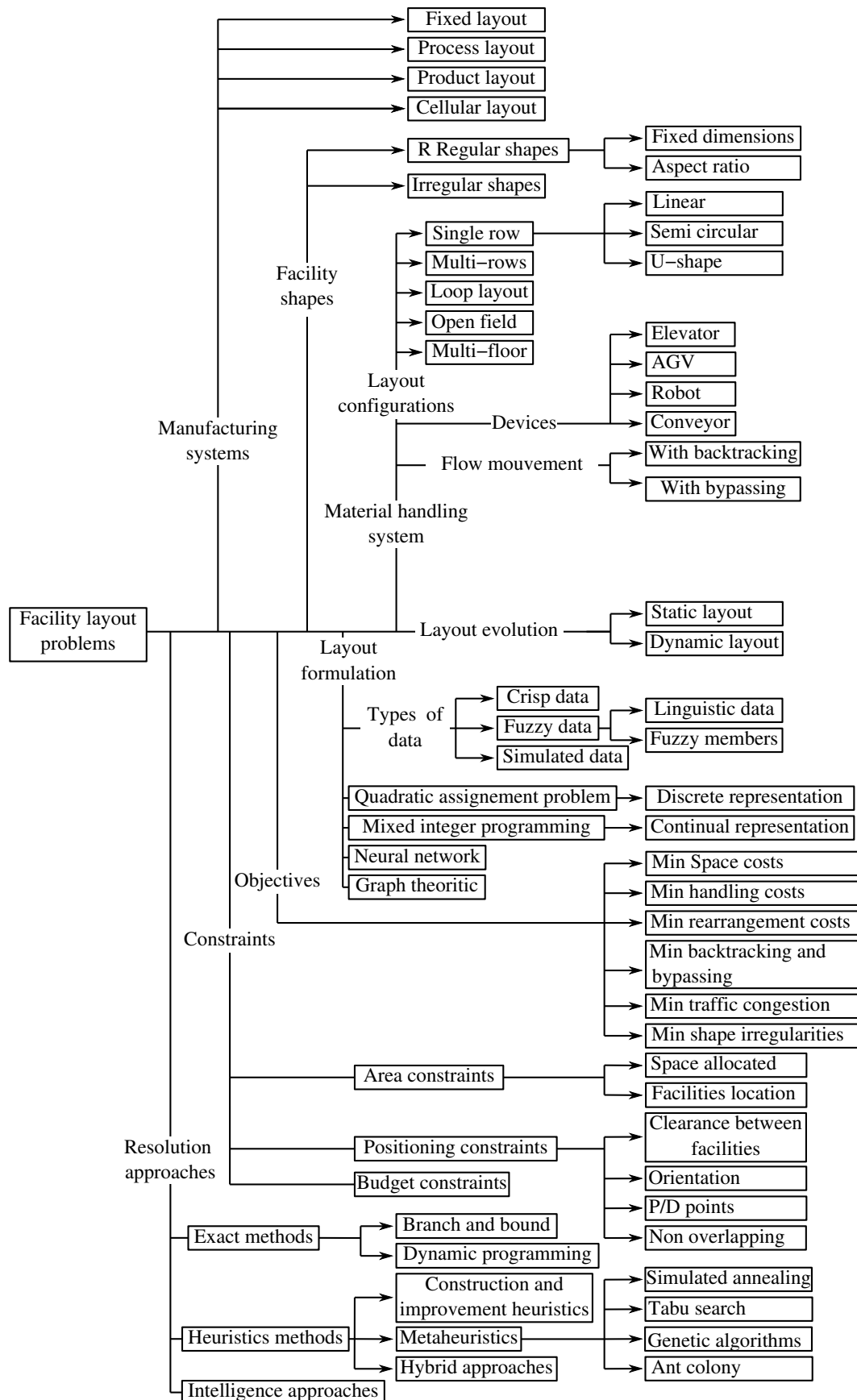


Figure 1.14 – Structure des problèmes d’agencement d’ateliers. Image empruntée à Drira *et al.* [DPHG07]

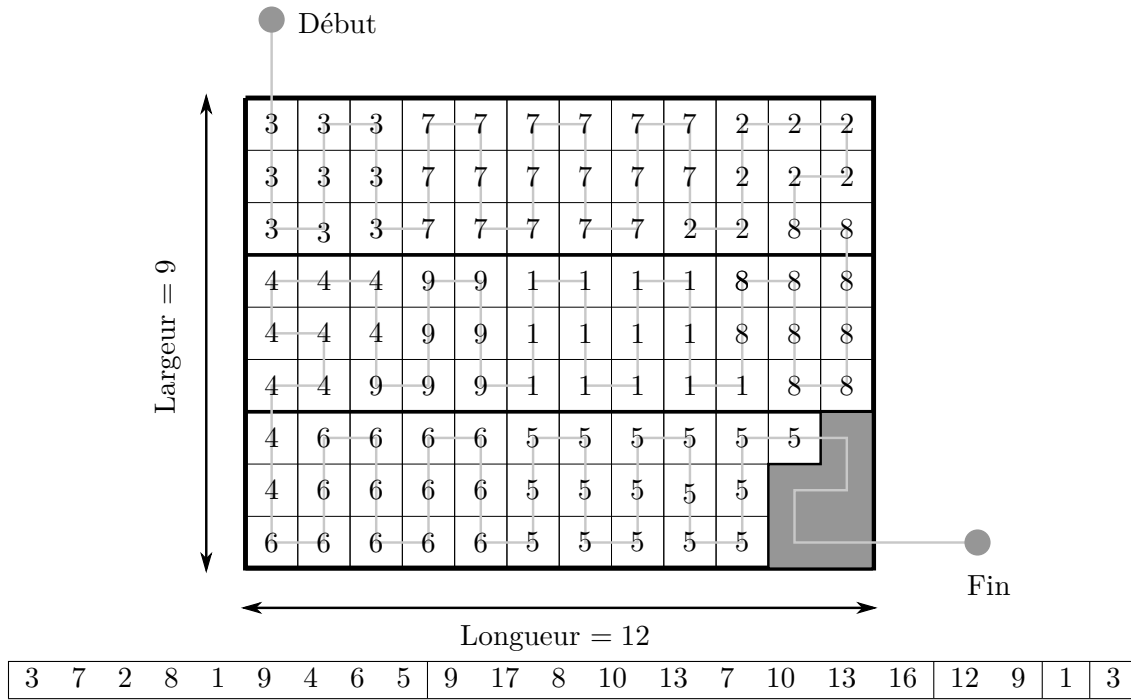


Figure 1.15 – Utilisation des *Space Filling Curves* proposées par Wang *et al.* [WHK05]. Les données sont représentées de la manière suivante : le premier segment représente la séquence d'introduction des locaux. Le second segment donne l'aire occupée par chaque local. Le troisième segment contient les dimensions du site (Longueur, Largeur). Le quatrième segment donne la direction de balayage (1 : horizontal, 2 : vertical). Le dernier segment indique la largeur de chaque bande.

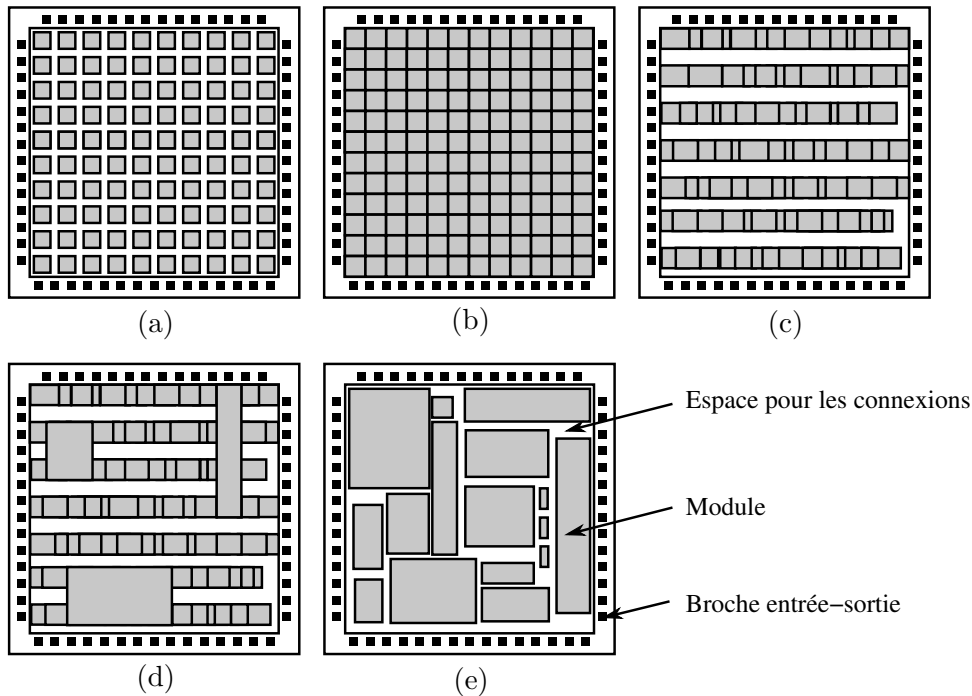


Figure 1.16 – Présentation des différents problèmes de placement de composants pour l'intégration de composants à grande échelle. (a) Gate array. (b) Sea-of-gates. (c) Standard-cells. (d) Mixed cell layout. (e) General-cell layout. Image reproduite de Egeblad [Ege03].

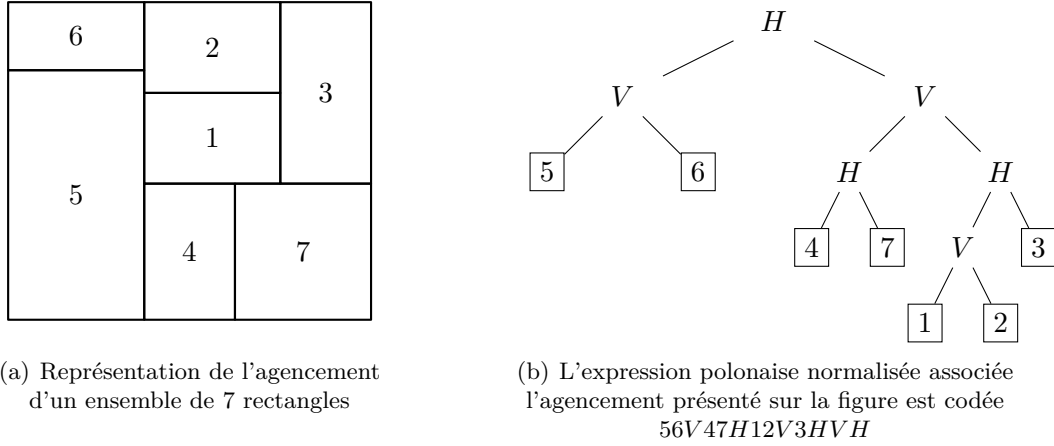


Figure 1.17 – Exemple d'agencement d'un ensemble de 7 rectangles avec sa notation polonaise normalisée correspondante.

1.5.2 Problèmes d'agencement tridimensionnel

Les problèmes d'agencement tridimensionnel se retrouvent principalement dans les problèmes d'ingénierie, où des systèmes complexes et compacts sont élaborés. Ils sont caractérisés par des composants de géométrie complexe, des formulations d'objectifs difficiles, des temps de calculs longs liés à l'évaluation des objectifs et des contraintes. En effet, les objectifs peuvent être le résultat de simulations physiques coûteuses (calculs éléments finis, ...). La géométrie des composants peut aussi entraîner des temps de calculs longs. Dans la majorité des exemples rencontrés, une méthode de placement relaxé est utilisée, et la géométrie des composants est approchée à l'aide des techniques d'*octrees*.

La résolution d'agencement tridimensionnel demande la maîtrise de la formulation des objectifs, des aspects géométriques ainsi que des méthodes d'optimisation. Cagan *et al.* [CSY02] ont proposé un état de l'art présentant l'ensemble de ces aspects. Ils notent que les problèmes sont traités comme des problèmes d'optimisation mono-objectif non contraint, où objectifs et contraintes sont agrégés dans une seule et même fonction. De nombreux exemples d'agencement 3D sont traités dans la littérature avec des méthodes spécifiques. Parmi ces exemples, on peut citer :

- l'agencement des différents éléments d'une pompe à chaleur [CCD⁺96] (Figure 1.2) ;
- la répartition des différents composants d'un moteur de voiture [Yin00] ;
- le chargement automatisé d'un volume [DC03] ;
- l'agencement des différents composants d'une transmission mécanique [YCH04] ;
- le positionnement des différents composants d'un satellite [TSqLL01, ZTS08].

Toutefois, aucun de ces exemples ne présente une méthode de résolution multi-objectif pouvant intégrer tout type de contraintes et objectifs.

1.5.2.1 Méthodes de résolution

On présente ici deux méthodes de résolution pour les problèmes d'agencement tridimensionnel.

Cagan *et al.* [YC00] ont développé une méthode de résolution relaxée basée sur les méthodes de recherche directe [Box57, HJ61, TT97]. Ces méthodes mono-objectif font évoluer un ensemble de solutions de proche en proche à l'aide de motifs pour identifier un minimum local, comme pour l'algorithme du simplexe de Nelder et Meade [NM65]. Pour connaître les directions d'évolution des solutions, ils construisent des fonctions d'influence qui leur permettent d'estimer les modifications qu'un changement de solutions va apporter. Initialement basée sur les modifications générées sur la fonction objectif (Algorithme *OPS*) [ACS07b], la fonction d'influence est maintenant calculée à partir des géométries des composants (Algorithme *MOPS*) [ACS07a]. Les résultats obtenus par les deux méthodes sont équivalents, toutefois l'exécution de l'algorithme *MOPS* est plus rapide que celle l'algorithme *OPS*, car le prétraitement lié à la construction des fonctions d'influence est beaucoup plus efficace.

Miao *et al.* [MFG08] utilisent l'algorithme génétique *NSGA-II* avec gestion de contraintes [DAPM00b] pour placer une douzaine de composants principaux sur un camion militaire : à savoir trois essieux, le moteur, le système de transmissions, le réservoir, le carburateur, la batterie, ... Le problème traité présente trois objectifs, qui concernent l'accessibilité, la sécurité, et la tendance au renversement. La connaissance associée au problème leur a permis de construire un modèle paramétré, sur lequel une optimisation multi-objectif a été réalisée. Les intersections entre composants sont évaluées à l'aide du logiciel de CAO ACIS¹. Les variables de positionnement sont réelles, l'orientation des composants est discrète. La méthode *SBX* est utilisée pour générer les croisements et un opérateur de mutation polynomiale est utilisée. Un codage binaire sur deux bits est utilisé pour représenter les quatre rotations de 90° d'un composant autour d'un axe. Un opérateur de croisement à un point (*Single point crossover*) est utilisé, et un échange de bits est utilisé comme opération de mutation. Comme l'ont remarqué Grignon et Fadel [GF04], l'utilisation de repère relatif permet de réduire l'espace de recherche et de limiter le nombre d'intersections possibles entre composants.

1.6 Comparaison des problèmes C&P et des problèmes d'agencement

La table 1.2 résume les différences entre les différents problèmes de placement. Les deux colonnes de cette table correspondent respectivement aux problèmes C&P ainsi qu'aux problèmes d'agencement. Chaque ligne met en avant une caractéristique d'un des deux problèmes.

1.7 Conseils pour la résolution d'un problème de placement

La première chose à faire consiste à déterminer s'il s'agit d'un problème de découpe et de conditionnement ou d'un problème d'agencement. Ce sont les objectifs et contraintes du problème qui renseignent l'utilisateur sur le type de problèmes. Pour résumer, dès qu'un des objectifs ou contrainte fait apparaître une interaction entre composants, le problème pourra être catégorisé comme problème d'agencement.

Dans le cas d'un problème de découpe, il y a de fortes chances que le problème appartienne à une catégorie répertoriée ci-dessus et que des méthodes de modélisation et résolution soient disponibles. On peut s'aider des classifications de Dyckhoff et Wäscher pour identifier le problème, ou bien encore utiliser la table 1.2 qui montre les différences entre chaque problème. Si les composants et le contenant sont de géométries régulières, il se peut qu'une méthode exacte garantissant l'optimalité du résultat soit disponible. En revanche, si le nombre de composants devient trop important ou que les géométries sont complexes, les méthodes de résolution seront basées des algorithmes stochastiques. Pour un même problème, les approches de résolution peuvent être totalement différentes : un problème de découpe de formes irrégulières peut être traité avec une méthode légale ou bien avec une méthode relaxée.

Dans le cas d'un problème d'agencement, les objectifs et les contraintes déterminent la nature du problème. À partir des données du problème, il faut :

- choisir un schéma de placement en fonction de la géométrie, des objectifs et des contraintes. Pour les problèmes d'agencement présentant des contraintes particulières comme l'accessibilité, seule une méthode de placement relaxé pourra être envisagée.
- choisir un jeu de variables d'optimisation adapté au problème, permettant d'intégrer un maximum de contraintes dans la modélisation du problème.

Si la compacité du problème est importante, il peut être difficile d'identifier des solutions initiales réalisables. Il est alors judicieux d'utiliser une heuristique de placement pour trouver un ensemble de solutions réalisables.

Le choix de la représentation géométrique des composants va être dicté par la géométrie et les degrés de libertés des composants, comme nous le verrons dans le chapitre 4.

1. ACIS : www.spatial.com/products/acis.html

PROBLÈMES DE PLACEMENT	
PROBLÈMES C&P	PROBLÈMES D'AGENCEMENT
Les composants sont fonctionnellement indépendants les uns des autres.	Les composants sont fonctionnellement et géométriquement reliés entre eux.
La compacité est toujours l'objectif sous-jacent.	La compacité peut ne pas être un objectif du problème.
Tous les composants ont le même type d'attributs.	Les composants n'ont pas nécessairement les mêmes types d'attributs.
Les problèmes C&P sont principalement mono-objectifs. Les problèmes multi-objectifs impliquent des types identiques d'objectifs.	Les problèmes d'agencement sont aussi bien des problèmes mono-objectifs que multi-objectifs.
Objectif(s) et contrainte(s) peuvent toujours être exprimés comme des fonctions mathématiques.	Objectif(s) et contrainte(s) peuvent ne pas s'exprimer comme des fonctions mathématiques. Ils peuvent être des valeurs données par le concepteur.
Les problèmes C&P prennent uniquement en compte les objectifs et contrainte(s) définis globalement.	Les problèmes d'agencement peuvent contenir des objectifs et des contraintes de différents types : individuel, interaction, global.
Les objectifs peuvent toujours s'exprimer comme une maximisation sur les composants ou une minimisation le contenant [WHS07].	Les objectifs ne présentent pas de caractéristiques particulières.
Les objectif(s) et contrainte(s) impliquent les attributs de tous les composants.	Objectif(s) et contrainte(s) peuvent impliquer les attributs de tous les composants, mais pas nécessairement.
Les problèmes C&P présentent uniquement des contraintes de non-chevauchement et d'appartenance.	Les problèmes d'agencement possèdent des contraintes de non-chevauchement et d'appartenance, mais peuvent aussi avoir des contraintes spécifiques (Alignement, Mise à distance, ...).
Les méthodes de résolution des C&P ne permettent pas de résoudre les problèmes d'agencement.	Les méthodes de résolution des problèmes d'agencement peuvent être utilisées pour la résolution des problèmes C&P.
L'évaluation des objectifs des C&P est quasi-immédiate.	L'évaluation des objectifs des problèmes d'agencement peuvent être coûteuses et nécessitent de choisir des stratégies de recherche appropriées.

Table 1.2 – Comparaison des problèmes C&P et des problèmes d'agencement.

1.8 Conclusion

Les problèmes de placement sont connus depuis bien longtemps, mais la recherche actuelle montre que ces problèmes ne sont entièrement pas résolus. Ils sont difficiles en bien des points. Pour les problèmes C&P, les efforts de recherche se portent actuellement sur l'amélioration des méthodes de résolution, pour obtenir de meilleurs résultats en moins de temps sur des instances toujours plus grandes. Concernant les problèmes d'agencement, les efforts doivent autant se concentrer sur la définition et la formalisation des problèmes que sur les méthodes de résolution.

Les méthodes d'optimisation utilisées pour la résolution des problèmes de placement sont présentées dans le chapitre suivant.

Chapitre 2

Modélisations et algorithmes d'optimisation pour les problèmes de placement

*Dans ce chapitre, les différentes méthodes et algorithmes
d'optimisation sont présentés de manière générale. L'accent est mis
sur les techniques d'optimisation qui seront utilisées dans la méthode
que nous présentons.*

Sommaire

2.1	Introduction	26
2.2	Optimisation multi-objectif	27
2.2.1	Méthodes de résolution des problèmes multi-objectifs	31
2.3	Algorithmes évolutionnaires	31
2.3.1	Algorithmes génétiques	33
2.3.2	Algorithmes génétiques utilisés	37
2.3.3	Discussion	38
2.4	Évaluation de la qualité des surfaces de compromis	38
2.4.1	Indicateurs unaires	38
2.4.2	Indicateurs binaires	41
2.4.3	Discussion	42
2.5	Conclusion	42

2.1 Introduction

L'état de l'art des problèmes de placement montre que l'ensemble des techniques d'optimisation ont été utilisées pour leur résolution. Collette et Siarry [CS02] ont regroupé sur un même graphique l'ensemble des méthodes existantes pour résoudre les problèmes d'optimisation. La figure 2.1 montre que les problèmes d'optimisation sont classés en deux catégories : les problèmes d'optimisation combinatoire et les problèmes d'optimisation continue. Une troisième catégorie mêlant ces deux types d'optimisations existe aussi. Les problèmes combinatoires se distinguent par le nombre fini de solutions. Les méthodes exactes sont utilisées pour la résolution des problèmes C&P avec des composants de géométrie rectangulaire ou parallélépipédique. Pour les grandes instances de ces problèmes ou lorsque les composants sont de géométrie complexe, les méthodes de résolution approchées sont utilisées. Ces méthodes sont principalement des métaheuristiques comme les algorithmes évolutionnaires ou encore des méthodes dédiées spécifiques à chaque problème. L'optimisation continue est utilisée lorsque l'espace de recherche est continu. Pour les problèmes linéaires mono-objectifs, la programmation linéaire est utilisée. Elle est généralement utilisée pour trouver des solutions relaxées pour les problèmes C&P. Lorsque la fonction objectif est non linéaire, différentes méthodes ont été proposées. La majorité est basée sur l'utilisation de gradient pour identifier l'optimum le plus rapidement. Dans les problèmes d'agencement, les gradients sont coûteux à obtenir et l'espace de recherche est multi-modal. Par conséquent, des métaheuristiques sont utilisées pour explorer l'espace de recherche. Enfin, la combinaison de métaheuristiques avec des méthodes d'optimisation locale forme les méthodes hybrides. Ces dernières exploitent au mieux les avantages des deux méthodes. La méthode que nous proposons chapitre 4 est une méthode hybride, basée sur un algorithme génétique et une optimisation locale.

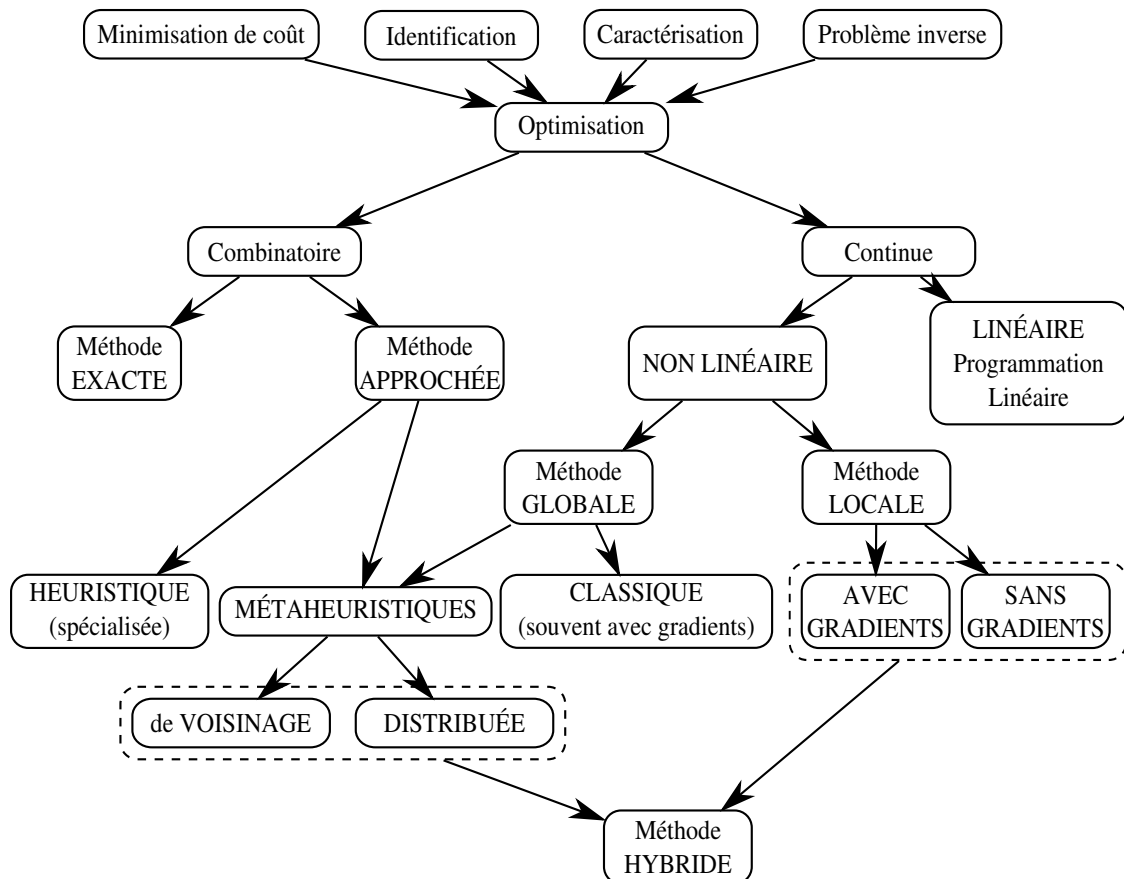


Figure 2.1 – Classification générale des méthodes d'optimisation. Figure extraite du livre de Collette et Siarry [CS02].

Dans la majorité des cas, les problèmes de placement sont formulés mathématiquement comme des problèmes d'optimisation. On peut toutefois signaler que d'autres formulations basées sur la programmation par contraintes existent. Ce genre d'approche a pour objectif de proposer l'ensemble des

solutions réalisables, c'est-à-dire toutes les solutions qui vérifient les contraintes spécifiées par l'utilisateur [BCP08].

2.2 Optimisation multi-objectif

Les problèmes de placement sont rarement des problèmes mono-objectifs. La plupart du temps, ces problèmes ont plusieurs objectifs contradictoires. Cette section présente les aspects théoriques de l'optimisation multi-objectif.

Dans sa forme la plus générale, un problème d'optimisation s'écrit mathématiquement :

$$(\mathcal{P}) \left\{ \begin{array}{l} \min_{\mathbf{x}} \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_p(\mathbf{x})) \\ \text{s.q.} \quad \begin{cases} g_j(\mathbf{x}) \leq 0 & \forall j \in \{1, \dots, G\} \\ h_k(\mathbf{x}) = 0 & \forall k \in \{1, \dots, H\} \\ x_i^l \leq x_i \leq x_i^u & \forall i \in \{1, \dots, n\} \end{cases} \end{array} \right. \quad (2.1)$$

où $\mathbf{f} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_p(\mathbf{x}))$ représente les différents objectifs du problème, $\mathbf{x} = (x_1, x_2, \dots, x_n)$ désigne les variables d'optimisation. Le vecteur de variables \mathbf{x} peut contenir des variables continues, binaires, discrètes ou encore des permutations. Il comprend les variables de positionnement, ainsi que les variables des attributs. Chaque variable x_i est comprise entre deux bornes x_i^l et x_i^u . Les fonctions g_j et h_k expriment respectivement les contraintes d'inégalités et d'égalités du problème. Les contraintes de placement sont comprises dans ces fonctions et sont gérées de différentes manières suivant le type de modélisation choisi.

Dans le cadre de problèmes multi-objectifs, il n'y a pas une solution optimale unique, mais un ensemble de solutions optimales. La résolution de tels problèmes consiste à trouver l'ensemble de ces solutions optimales. Ces dernières sont appelées solutions efficaces. Dans cet ensemble, aucune des solutions n'est meilleure qu'une autre, aucune n'étant systématiquement inférieure aux autres sur tous les objectifs.

Un problème d'optimisation multi-objectif (\mathcal{MO}) consiste à minimiser simultanément un ensemble de p fonctions objectifs :

$$\min_{\mathbf{x} \in E} \mathbf{f}(\mathbf{x}) = \min_{\mathbf{x} \in E} (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_p(\mathbf{x})) \quad (2.2)$$

où E désigne l'espace des solutions réalisables. Chaque *solution* \mathbf{x} a pour image dans l'espace des objectifs un *point* $\mathbf{y} : \mathbf{y} = \mathbf{f}(\mathbf{x})$ avec $\mathbf{y} \in \mathbb{R}^p$.

Dans le contexte multi-objectif, on utilise le principe de dominance pour comparer deux points.

Définition 1 (Définition de la dominance)

Une solution $\mathbf{x} \in E$ domine $\mathbf{x}' \in E$ si elle vérifie

$$\begin{aligned} \forall i \in \{1, \dots, p\}, f_i(\mathbf{x}) &\leq f_i(\mathbf{x}') \\ \exists i \in \{1, \dots, p\}, f_i(\mathbf{x}) &< f_i(\mathbf{x}') \end{aligned} \quad (2.3)$$

On note cette relation de dominance $\mathbf{x} \preceq \mathbf{x}'$.

Cette définition permet d'introduire le concept de Pareto-optimalité.

Définition 2 (Définition de Pareto-optimal)

Une solution \mathbf{x} est dite efficace ou Pareto-optimale si elle n'est dominée par aucune autre solution appartenant à E . L'ensemble de ces solutions sont également appelées solutions non dominées.

Définition 3 (Front de Pareto) L'image des solutions efficaces forme dans l'espace des objectifs un ensemble de points non dominés, communément appelé front de Pareto.

La figure 2.2 regroupe l'ensemble des informations présentées pour un problème de minimisation bi-objectif.

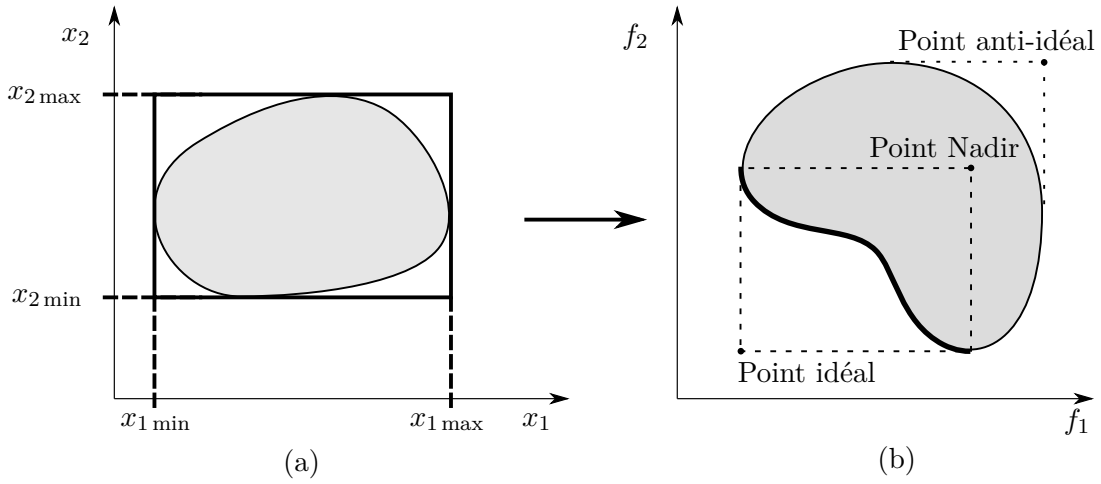


Figure 2.2 – Illustration des espaces des variables et des objectifs pour un problème de minimisation bi-objectif avec deux variables. Le domaine grisé de la sous-figure (a) représente l'ensemble des solutions réalisables E . Le trait épais noir de la sous-figure (b) représente le front de Pareto du problème, c'est-à-dire l'ensemble des points non dominés.

Pour certains problèmes difficiles, il peut être intéressant de relaxer la contrainte de dominance. Cette relaxation est définie au travers de l' ε -dominance.

Définition 4 (Définition de l' ε -dominance)

Une solution $\mathbf{x} \in E$ ε -domine $\mathbf{x}' \in E$ si elle vérifie

$$\begin{aligned} \forall i \in \{1, \dots, p\}, f_i(\mathbf{x}) &\leq f_i(\mathbf{x}') \\ \exists i \in \{1, \dots, p\}, f_i(\mathbf{x}) &< f_i(\mathbf{x}') - \varepsilon_i \end{aligned} \quad (2.4)$$

On note cette relation de dominance $\mathbf{x} \preceq_{\varepsilon} \mathbf{x}'$.

L'utilisation de l' ε -dominance permet d'accroître le nombre de points non dominés, et d'inclure des points inférieurs dans le front de Pareto. Les différents ε_i sont calculés à partir d'une valeur ε définie par le concepteur. Généralement, les ε_i sont évalués de la manière suivante :

$$\varepsilon_i = \varepsilon (f_i^{\max} - f_i^{\min}) \quad (2.5)$$

Idéalement, les valeurs de f_i^{\min} et f_i^{\max} sont issues des points idéal et Nadir. Le point idéal est défini à partir des valeurs minimales que peuvent prendre l'ensemble des points de l'espace d'arrivée, et le point Nadir est calculé à partir des valeurs maximales des objectifs de l'ensemble des points non dominés. Ce point ne doit pas être confondu avec le point anti-idéal, obtenu pour les pires valeurs de l'espace d'arrivée. (Figure 2.2). Classiquement, dans les métaheuristiques à population, ces valeurs sont obtenues en cherchant le minimum et maximum sur chacun des objectifs. Dans le cas où $\varepsilon = 0$, l'utilisation de l' ε -dominance revient à utiliser la dominance classique. Si $\varepsilon = 1$, alors toutes les points sont inclus dans le front de Pareto, aucune solution ne domine aucune autre solution.

Les sous-figures 2.3(a) et 2.3(b) présentent respectivement la relation de dominance classique et la relation d' ε -dominance. Sur ces deux sous-figures, le point A divise l'espace bi-objectif en trois zones : une zone où les points sont dominés par A (zone en gris foncé), une zone où les points dominent A (zone en gris clair) et une zone complémentaire où les points ne sont ni dominés ni dominant par rapport à A . Le point A domine le point E , mais est dominé par B . Les points C et D ne sont pas dominés par A et A ne domine pas ces points. Ces trois points sont dits incomparables. La sous-figure (b) illustre la notion d' ε -dominance, utilisée pour relaxer le critère de dominance. Les points situés dans le rectangle autour de A lui seront équivalents. Lorsque l'ensemble des composantes du vecteur ε tend vers zéro, l' ε -dominance est équivalente à la dominance classique.

On peut aller plus loin en introduisant les définitions suivantes :

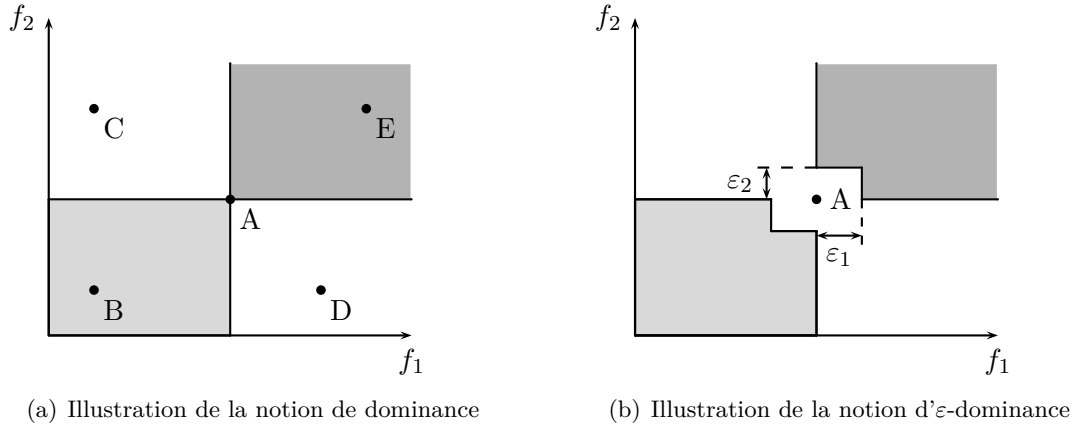


Figure 2.3 – Illustration de quelques-unes des notions de dominance pour un problème de minimisation bi-objectif.

Définition 5 (Définition de dominance stricte) Une solution $\mathbf{x} \in E$ domine strictement $\mathbf{x}' \in E$ si elle vérifie : $f_i(\mathbf{x}) < f_i(\mathbf{x}'), \forall i \in \{1, \dots, p\}$. On note cette relation de dominance stricte $\mathbf{x} \prec \mathbf{x}'$.

Définition 6 (Définition du front de Pareto faible) L'ensemble des solutions faiblement non dominées d'un ensemble P est l'ensemble des solutions de cet ensemble P qui ne sont pas strictement dominées. Cet ensemble définit le front de Pareto faible.

La figure 2.4 présente le front de Pareto et le front de Pareto faible pour un problème de minimisation bi-objectif. La figure 2.5 présente pour un problème bi-objectif le front de Pareto en fonction des différentes combinaisons d'objectifs. Ce nombre est égal à 2^p , soit 4 ici.

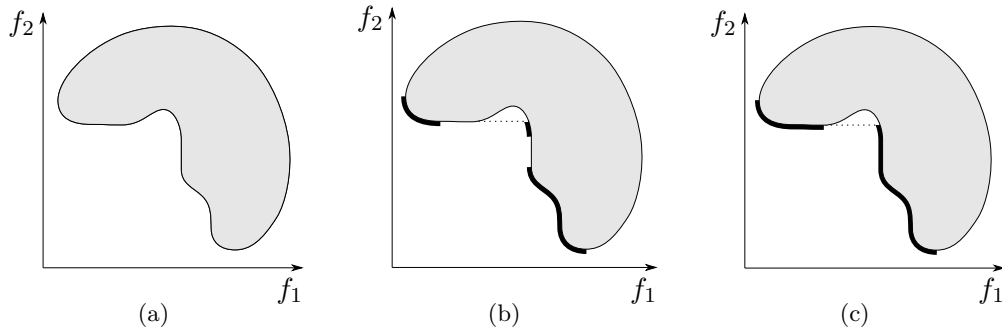


Figure 2.4 – Front de Pareto et front de Pareto faible. (a) représente l'espace d'arrivée d'un problème de minimisation bi-objectif, pour lequel on cherche l'ensemble des solutions optimales. Le trait noir épais de la sous-figure (b) représente le front de Pareto du domaine. Sur la sous-figure (c), ce trait représente l'ensemble des solutions faiblement non dominées.

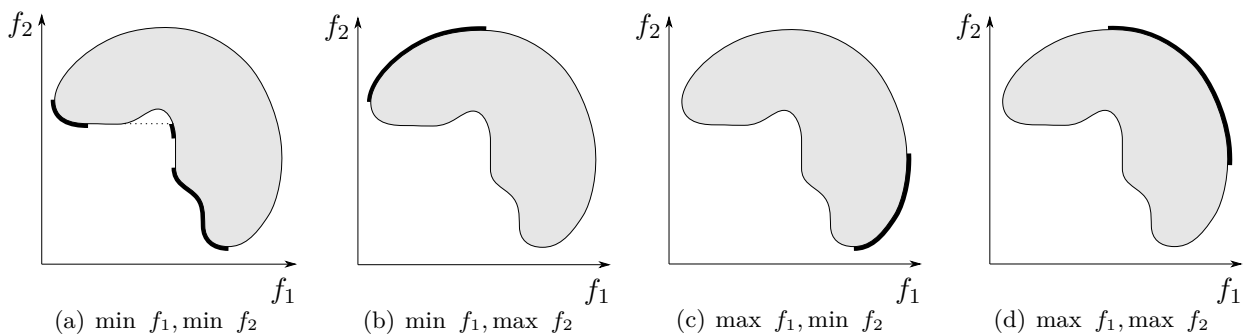


Figure 2.5 – Fronts de Pareto sur un problème bi-objectif pour les quatre combinaisons possibles des deux objectifs. L'espace d'arrivée est représenté en gris. Les lignes en trait épais représentent les fronts de Pareto.

La plupart des problèmes d'optimisation présentent des contraintes. Ces dernières sont utilisées pour distinguer les solutions réalisables et non réalisables. Classiquement, les valeurs des contraintes non respectées sont sommées dans un indice de violation de contraintes. Il est important que les valeurs des contraintes aient le même ordre de grandeur, de manière ne pas privilégier une contrainte particulière. Cet indice permet à l'aide d'une seule valeur de savoir si les contraintes sont respectées. Il est utilisé pour définir une notion de contrainte-dominance.

Définition 7 (Définition de la dominance pour un problème contraint) Une solution \mathbf{x} contrainte-domine une solution \mathbf{x}' , ($\mathbf{x} \preceq_c \mathbf{x}'$), si l'une des trois conditions suivantes est vérifiée :

1. la solution \mathbf{x} est réalisable alors que la solution \mathbf{x}' ne l'est pas ;
2. les solutions \mathbf{x} et \mathbf{x}' sont toutes les deux irréalisables, mais la solution \mathbf{x} présente un indice de violation de contraintes inférieur ;
3. les solutions \mathbf{x} et \mathbf{x}' sont toutes les deux réalisables, et la solution \mathbf{x} domine la solution \mathbf{x}' au sens classique.

Dans un problème multi-objectif contraint, si l'ensemble des solutions est non réalisable, l'optimisation sera une optimisation mono-objectif dont l'objectif sera la minimisation des indices de violations de contraintes.

Pour classer les individus d'un problème multi-objectif, on utilise la notion de rang. Le rang traduit la position d'un individu dans une population. Il existe plusieurs définitions du rang dans la littérature multi-objectif, les plus connues étant celle de Goldberg et celle de Fonseca et Fleming :

- Rang de Goldberg [Gol89] : Le rang d'un individu est défini comme le nombre de fronts le dominant plus 1. Les individus du front de Pareto ont un rang égal à un. Le reste des individus ont alors un rang supérieur ou égal à 2. Ce rang est similaire au rang défini dans la théorie des ensembles ordonnés, à la différence que celui-ci débute à 0. Les algorithmes *NSGA-II* et *Omni-Optimizer*, qui seront utilisés par la suite, utilisent cette définition de rang.
- Rang de Fonseca et Fleming [FF93] : Le rang d'un individu est alors défini comme le nombre d'individus dominés plus 1. C'est le rang utilisé dans l'algorithme *MOGA* [FF93].

La figure 2.6 présente une comparaison de ces rangs.

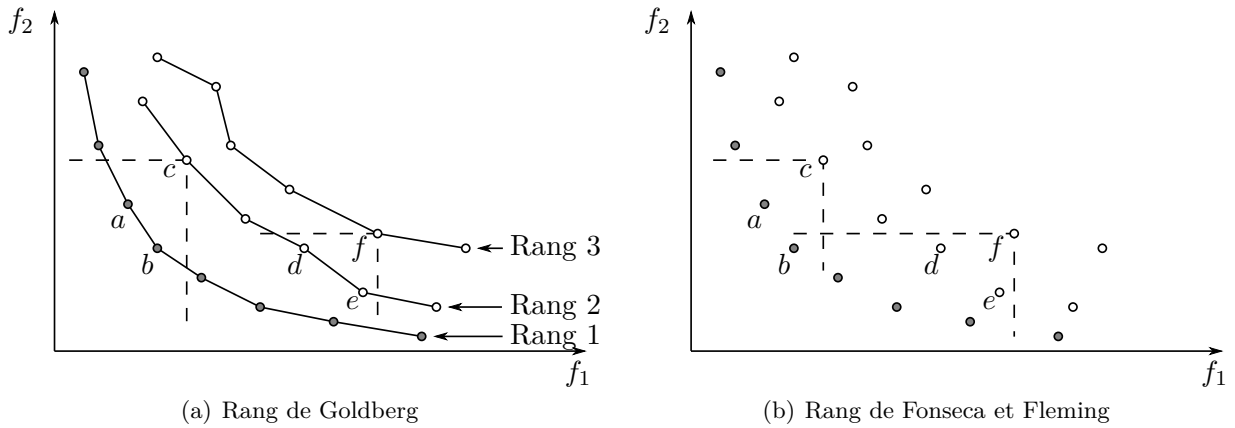


Figure 2.6 – Illustration des différences entre le rang de Goldberg et le rang de Fonseca et Fleming pour un problème de minimisation bi-objectif. Chaque cercle représente un individu dans l'espace des objectifs. Les points grisés représentent les solutions de rang 1. Quel que soit le rang utilisé, les points non dominés ont un rang égal à 1, ils représentent les solutions du front de Pareto. La sous-figure (c) permet de comparer les différences des rangs pour les points a , b , c , d , e et f .

Pour les problèmes multi-objectifs, une formulation multi-objectif présente de nombreux avantages :

1. elle permet de balayer l'espace de conception ;
2. elle permet d'obtenir l'ensemble des solutions non dominées et pas uniquement les solutions non dominées de l'enveloppe convexe de l'espace d'arrivée ;
3. la méthode permet de choisir une solution parmi plusieurs solutions proposées. Des outils d'aide à la décision permettent de faire ce choix à partir de critères définis sur les variables, les fonctions objectif et de préférences définies entre les solutions.

2.2.1 Méthodes de résolution des problèmes multi-objectifs

De nombreuses méthodes ont été proposées pour la résolution des problèmes multi-objectifs. Ces méthodes peuvent être catégorisées dans deux familles : les approches non-Pareto et les approches Pareto. Les approches non-Pareto se ramènent classiquement à une optimisation mono-objectif, pour laquelle les méthodes de résolution sont connues depuis longtemps. Les approches Pareto qui ne transforment pas les objectifs du problème, sont présentées par le biais des algorithmes évolutionnaires, algorithmes stochastiques faisant l'objet de la section suivante.

Parmi les approches non-Pareto, la méthode des sommes pondérées est sans doute la plus utilisée de par sa simplicité de mise en œuvre. Agrégeant l'ensemble des fonctions dans une seule et même fonction coût, elle permet de se ramener à un problème d'optimisation mono-objectif.

$$\min_{x \in E} \sum_{i=1}^p w_i f_i(x) \quad (2.6)$$

La simplicité d'agrégation des différents objectifs a largement contribué au succès de cette méthode.

Dans la plupart des cas, les poids sont fixés arbitrairement par l'utilisateur. Toutefois, il existe différentes méthodes pour déterminer les poids, notamment la méthode de comparaison par paires. En choisissant des poids tels que $w_i = 1$ et $w_j = 0, \forall j \neq i$, cela revient à faire une optimisation lexicographique. La variation des différents poids w_i permet de balayer uniquement la partie convexe de la frontière de Pareto : seules les solutions supportées sont obtenues (Théorème de Geoffrion [Geo68]).

Le problème suivant, présenté par Deb [Deb01], permet d'illustrer ce problème. Il permet de faire varier la géométrie du front du Pareto à l'aide de deux paramètres réels a et b .

$$(P) \begin{cases} \min f_1 = x_1 \\ \min f_2 = 1 + x_2^2 - x_1 + a \sin(b\pi x_1) \\ \text{s.c.} \quad \begin{cases} 0 \leq x_1 \leq 1 \\ -2 \leq x_2 \leq 2 \end{cases} \end{cases} \quad (2.7)$$

Les figures 2.7 et 2.8 illustrent la différence de résultats pour la méthode des sommes pondérées avec un espace d'arrivée convexe et non convexe. La figure 2.7 présente un front de Pareto convexe. L'ensemble des solutions efficaces associé à ce front peut être identifié à l'aide de la méthode des sommes pondérées. En revanche, la figure 2.8 présente un front concave, pour lequel seuls les points supportés du front de Pareto (points noirs) peuvent être identifiés.

Bien d'autres méthodes se ramenant à un problème d'optimisation mono-objectif ont été présentées dans la littérature. Les plus connues sont la méthode des ε -contraintes, la méthode Min-Max, la méthode du but à atteindre [Gem74] ou encore la méthode de Benson [Ben78, Ehr00]. Ces méthodes moins répandues que la méthode des sommes pondérées permettent d'identifier des fronts de Pareto non convexe.

2.3 Algorithmes évolutionnaires

Les algorithmes évolutionnaires, algorithmes évolutionnistes ou le calcul évolutionnaire (*Evolutionary Algorithms, EA*), sont une famille de métaheuristiques stochastiques s'inspirant de la théorie de l'évolution.

Les algorithmes évolutionnaires regroupent une large palette de familles d'algorithmes, parmi lesquelles on distingue :

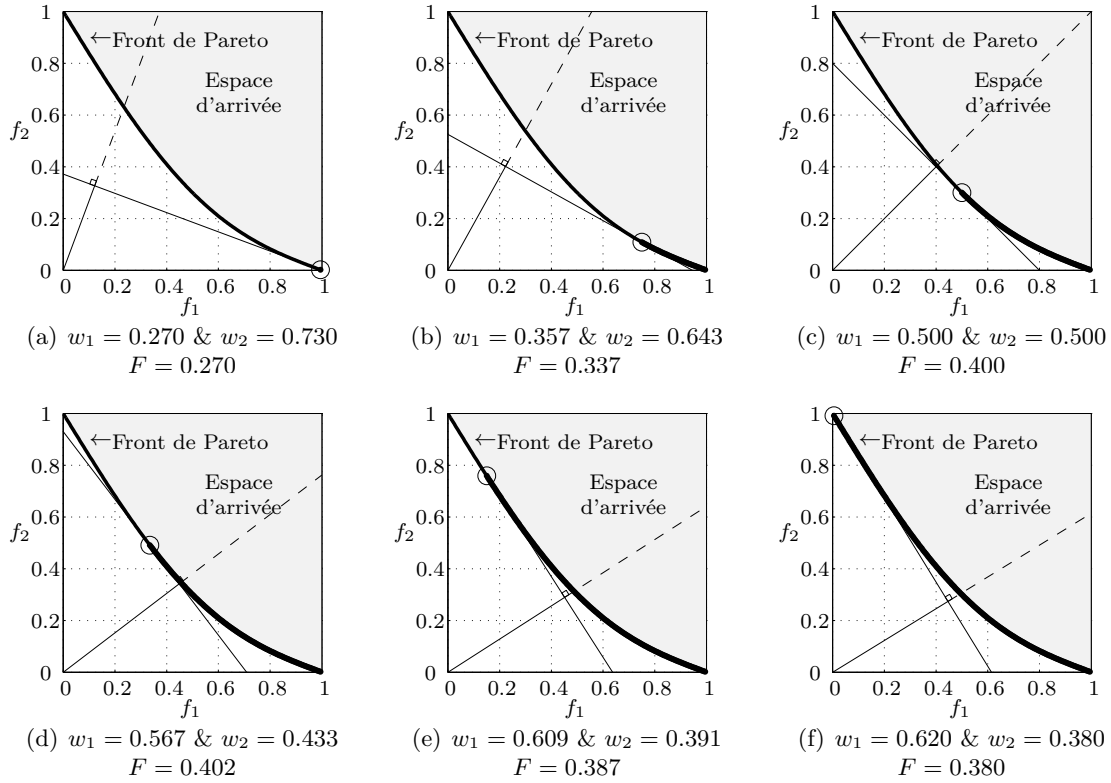


Figure 2.7 – Illustration de la méthode des sommes pondérées pour un espace d'arrivée convexe. L'ensemble des points de la figure représente le front de Pareto associé au problème 2.7 avec $a = 0.2$ et $b = 1$. Lorsque le front de Pareto est un ensemble convexe, la méthode des sommes pondérées permet d'identifier l'ensemble des points non dominés.

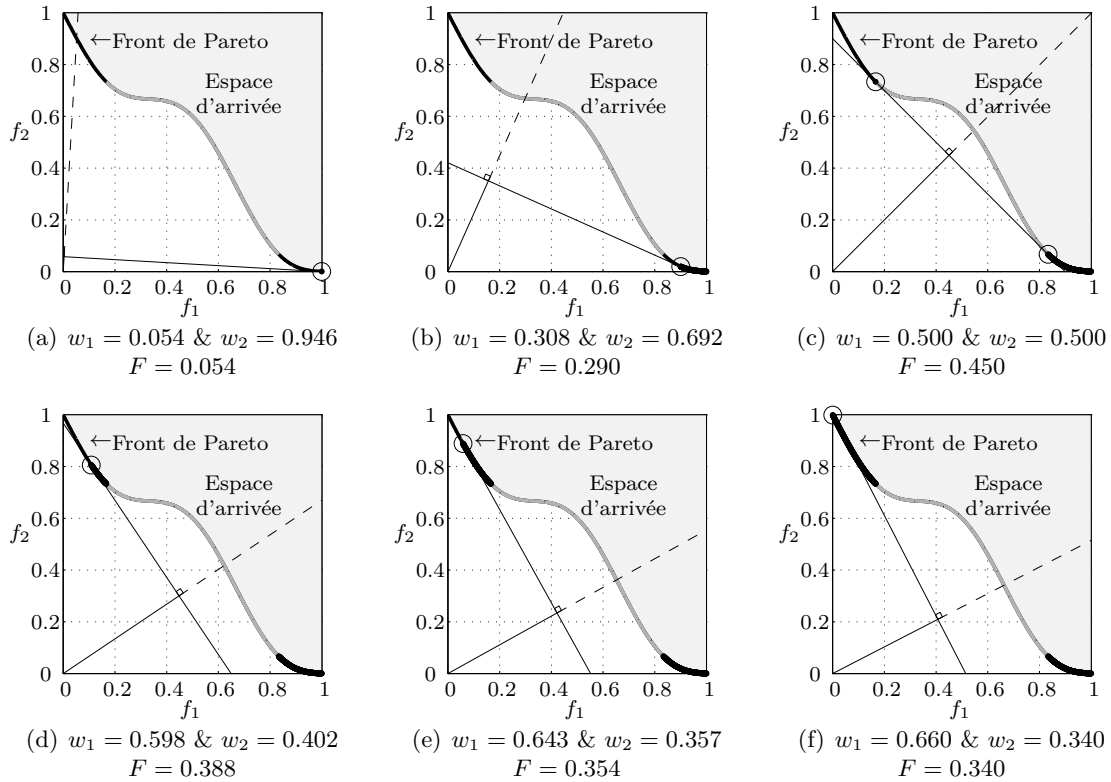


Figure 2.8 – Illustration de la méthode des sommes pondérées pour un espace d'arrivée non convexe. Cette illustration présente le problème lié à l'utilisation de l'approche des sommes pondérées pour l'optimisation multi-objectif. L'ensemble des points noirs et gris de la figure représente le front de Pareto associé au problème 2.7 avec $a = 0.1$ et $b = 3$.

- les stratégies d'évolution (*ES*). Basées sur des opérateurs de sélection et de mutation, ce sont les premiers algorithmes évolutionnaires. Depuis leurs apparitions en 1965, plusieurs variantes ont été développées, parmi lesquelles $(1 + 1) - ES$, $(\mu + \lambda) - ES$, [Rec65, Sch68, Rec73] ainsi que *PAES* [KC00], une version multi-objectif des stratégies d'évolution ;
- les algorithmes génétiques (*GA*). Les *GA* sont les plus connus des algorithmes évolutionnaires, ils sont présentés en détails ci-après [Deb01] ;
- l'évolution différentielle (*DE*) est une autre métaheuristique à population apparue en 1997 basée sur le concept de mutation vectorielle [PSL05] ;
- les algorithmes mémétiques (*MA*). Ce sont des algorithmes évolutionnaires hybrides utilisant une méthode de recherche locale en fin d'optimisation, pour atteindre l'optimum global.

Bien d'autres familles d'algorithmes évolutionnaires existent mais celles-ci vont bien au delà de la finalité de nos recherches. Les avantages des algorithmes évolutionnaires sont nombreux :

1. leur mise en œuvre est généralement simple,
2. ils sont robustes. Ils ne sont pas aussi sensibles que les méthodes d'optimisation déterministes,
3. ils permettent d'intégrer différents types de variables lors de l'optimisation,
4. les calculs peuvent être facilement parallélisables, contrairement à la plupart des autres méthodes,
5. ils permettent de traiter les problèmes multi-objectifs.

Pour toutes ces raisons, les algorithmes évolutionnaires suscitent un grand intérêt dans la communauté scientifique.

2.3.1 Algorithmes génétiques

Imitant le processus naturel de l'évolution, les algorithmes génétiques (*Genetic algorithms*, *GA*) sont des algorithmes évolutionnaires, faisant partie des métaheuristiques à population. Basés sur les principes de survie et de reproduction décrits par Charles Darwin [Dar59], les algorithmes génétiques cherchent à améliorer la population courante. Ils se décomposent généralement en deux phases : une phase exploratoire ou de recherche et une phase d'intensification.

Comme tous les algorithmes évolutionnaires, les algorithmes génétiques travaillent sur un ensemble de solutions, appelées individus. L'ensemble de ces individus forment une population et l'objectif des *GA* consiste à faire évoluer au fil des générations les individus vers un ou plusieurs optimaux globaux [Hol75]. Les algorithmes génétiques sont des algorithmes d'optimisation très robustes particulièrement adaptés aux problèmes où l'initialisation n'est pas intuitive, les variables du problème ne sont pas toutes du même type (réel, entier, booléen).

2.3.1.1 Principes des algorithmes génétiques

La figure 2.9 présente les différentes étapes d'un algorithme génétique générationnel, à savoir l'évaluation d'une population, la sélection d'individus pour la génération d'une nouvelle population et les opérations génétiques de croisement et de mutation. L'algorithme s'arrête dès lors qu'un critère d'arrêt est satisfait, comme par exemple un nombre maximum de générations, une détection de convergence du problème, ... Il existe plusieurs variétés d'algorithmes génétiques, à savoir les algorithmes mono-objectifs, multi-objectifs, générationnels (*generational GA*), les algorithmes à génération continue (*steady-state GA*). À la différence des algorithmes générationnels, les algorithmes à génération continue permettent d'utiliser les solutions nouvellement calculées pour générer de nouvelles solutions.

Initialement proposés par Holland [Hol75] et Goldberg [Gol89], les algorithmes génétiques ont rapidement évolué pour résoudre des problèmes multi-objectifs.

2.3.1.2 Les algorithmes génétiques multi-objectifs

L'objectif des algorithmes génétiques multi-objectifs est de proposer l'ensemble des solutions efficaces, dont les points sont non dominés. Ils sont basés sur le concept de dominance pour classer les solutions.

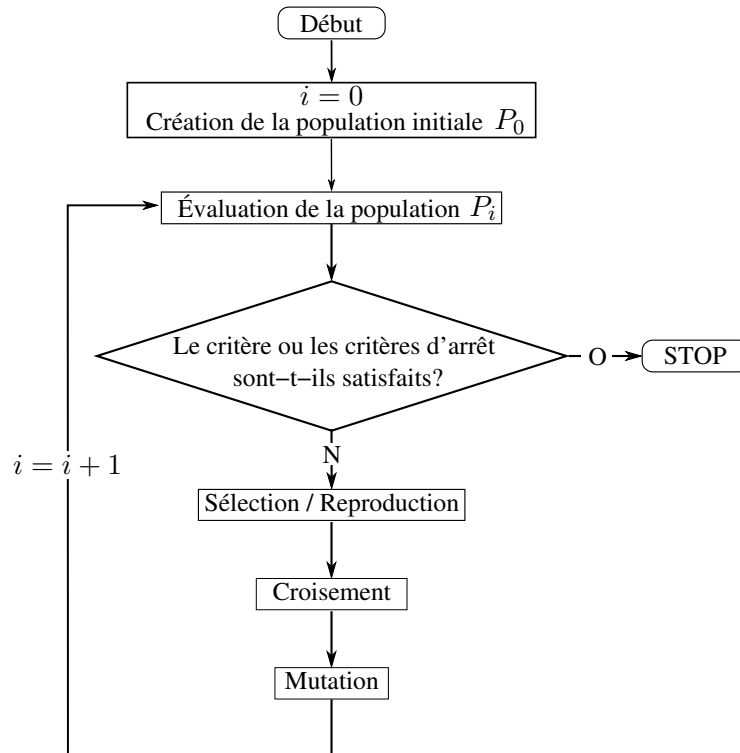


Figure 2.9 – Principe de fonctionnement des algorithmes génétiques.

Les algorithmes *NSGA-II* [DAPM00a, DAPM00b] et *SPEA2* [ZLT01] sont les algorithmes génétiques multi-objectifs de référence. La plupart des nouvelles métaheuristiques à population sont comparées à ces deux algorithmes. Toutefois, la littérature abonde d'algorithmes multi-objectifs, dont les premiers datent des années 1980 : *VEGA* [Sch84], *VOES* [Kur90], *RWGA* [MI95], *MOGA* [FF93], *NPGA2* [EMH01]. Les différences entre tous ces algorithmes résident dans la manière dont sont réalisées les diverses opérations génétiques, dont l'élitisme est pris en compte,...

2.3.1.3 Opérateurs génétiques

Les opérateurs génétiques sont l'ensemble des opérations réalisées sur les individus pour générer de nouveaux individus. Ces opérateurs sont très importants, car ce sont eux qui font évoluer les individus d'une génération à l'autre. Les performances des algorithmes génétiques dépendent grandement du choix de ces opérateurs et de leurs réglages. Les opérateurs génétiques utilisés vont dépendre du codage des variables d'optimisation.

2.3.1.3.1 Opérateurs de sélection et élitisme. L'objectif premier de l'opérateur de sélection est de sélectionner les individus à fort potentiel / de bonnes qualités et d'éliminer les autres tout en conservant la taille de la population. L'identification des bons individus se fait par des opérateurs de sélection. Plusieurs méthodes ont été élaborées, les plus connues étant la méthode de sélection proportionnelle aussi connue sous le nom de sélection par roulette. Cette dernière a été améliorée pour donner la méthode de sélection universelle stochastique. Encore bien d'autres méthodes existent, comme la sélection par tournoi qui utilise des comparaisons par paire de solutions pour sélectionner les meilleurs individus. C'est la technique la plus utilisée lors d'optimisation de problèmes sous contraintes, où l'ensemble des valeurs des contraintes est regroupé dans un indice de violation de contraintes.

Pour éviter de perdre les meilleures solutions identifiées, plusieurs opérateurs d'élitisme ont été proposés. En ne perdant aucune des meilleures solutions, l'élitisme permet d'améliorer considérablement la convergence des algorithmes génétiques. L'élitisme peut être implémenté de différentes manières :

dans l'algorithme *SPEA2*, l'ensemble des solutions efficaces est stocké dans une population archive, qui prend part à la création de nouvelles solutions. Dans l'algorithme *NSGA-II*, il est réalisé de manière explicite en sélectionnant les meilleures solutions des populations parent et enfant. La figure 2.10 illustre le fonctionnement de la procédure élitiste de l'algorithme *NSGA-II*. Une fois qu'une population enfant Q_t vient d'être évaluée, celle-ci est fusionnée avec sa population parent P_t . Cette nouvelle population R_t est triée selon le rang de Goldberg, et les solutions présentant les meilleurs rangs sont extraites pour former une nouvelle population parent P_{t+1} .

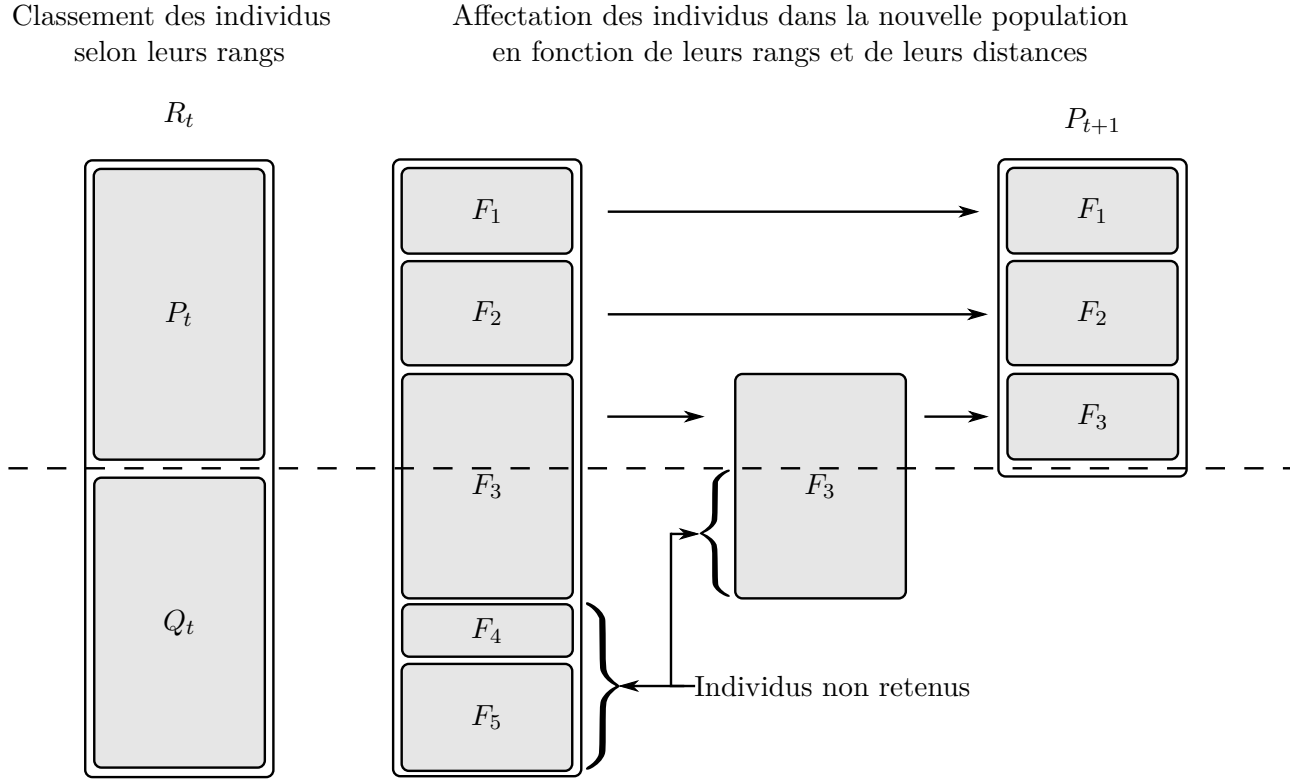


Figure 2.10 – Déroulement de la procédure élitiste de l'algorithme *NSGA-II*. Image reproduite de Deb [Deb01].

2.3.1.3.2 Opérateurs de croisement et de mutation. Les opérateurs de croisement et de mutation sont utilisés pour générer de nouvelles solutions à partir des solutions sélectionnées. Chaque opérateur est défini pour un type de variables particulier (réel, binaire, permutation,...). Il existe donc autant d'opérateurs qu'il y a de type de variables. Ces opérateurs agissent avec une probabilité définie par l'utilisateur, toute la difficulté d'utilisation des GA résidera dans le réglage de ces paramètres. En plus de ces opérateurs, l'utilisateur peut ajouter ses propres opérateurs, comme nous l'avons fait pour la méthode proposée chapitre 4. Les opérateurs génétiques de croisement et de mutation sur les variables réelles, qui seront utilisés par la suite, sont présentés en annexe page 153.

2.3.1.3.3 Opérateurs de diversité. À leurs débuts, les algorithmes génétiques cherchaient à générer un ensemble de solutions efficaces. Ces solutions étaient généralement concentrées autour d'un seul et même point. Pour éviter que tous les individus d'une même population ne convergent vers un seul et même individu, un mécanisme de préservation de diversité phénotypique est introduit [GR87]. Cet opérateur a pour objectif de proposer un ensemble de points répartis équitablement sur la frontière de Pareto, en calculant des distances entre les points. Classiquement, ce mécanisme génère une modification de la fonction d'efficacité fonction de la répartition des points dans l'espace des objectifs.

La figure 2.11 présente les deux techniques utilisées par les algorithmes *NSGA-II* et *SPEA2* pour prendre en compte la diversité phénotypique. *NSGA-II* calcule les distances phénotypiques entre l'ensemble des points non dominés et essaie d'uniformiser la distribution des points le long de la surface de compromis. *SPEA2* estime la densité de répartition des points regroupés en *clusters*.

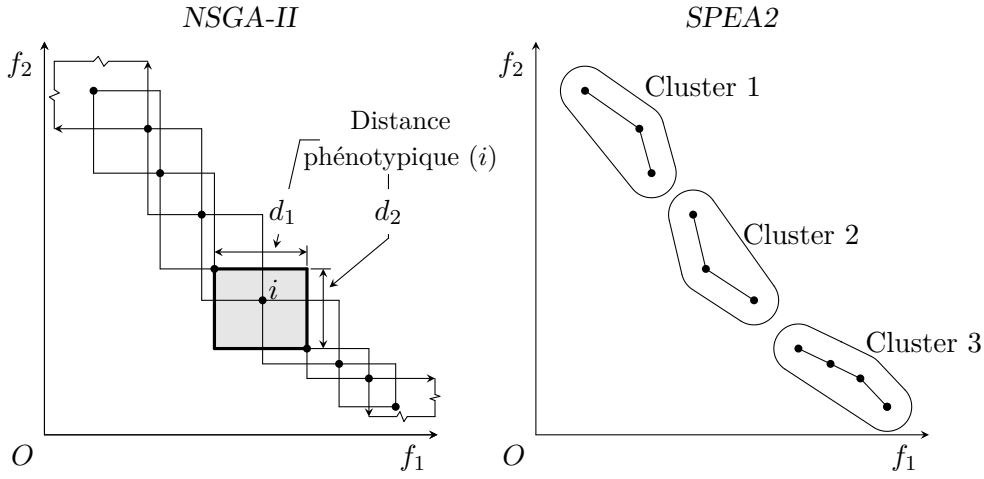


Figure 2.11 – Comparaison des mécanismes de diversité phénotypiques entre *NSGA-II* et *SPEA2*.

Toutefois, ces méthodes ne tiennent pas compte de la diversité dans l'espace des variables (diversité génotypique), qui peut être un élément de décision important pour le concepteur. La figure 2.12 présente le cas où des solutions sont très proches dans l'espace des objectifs mais très éloignées dans l'espace des variables. La majorité des algorithmes évolutionnaires s'intéresse majoritairement à la diversité dans l'espace des objectifs et pas ou peu à la diversité génotypique. Quelques algorithmes font mention de cette diversité et essaient de l'intégrer lors de la résolution des problèmes. Pour prendre en compte les diversités phénotypique et génotypique, Deb *et al.* [DT08] ont développé *Omni-Optimizer*, dans lequel ils ont introduit de nouveaux opérateurs : lors de la sélection des individus, les solutions sont comparées selon leur rang dans la population mais aussi selon les distances phénotypique et génotypique. La prise en compte de ces distances a pour objectif de conserver la diversité dans l'espace des objectifs et des variables. De plus, un opérateur de sélection restreinte a été introduit. Cet opérateur sélectionne au hasard deux solutions dans la population parent et les remplace par les deux solutions les plus proches dans l'espace des objectifs présentant un meilleur rang. Deb *et al.* [DT08] ont montré que cet opérateur permettait de trouver les solutions multi-modales d'un problème. Shir *et al.* [SPNE09] ont modifié la stratégie d'évolution *CMA-ES* [HMK03], pour intégrer la diversité génotypique lors de la sélection des individus.

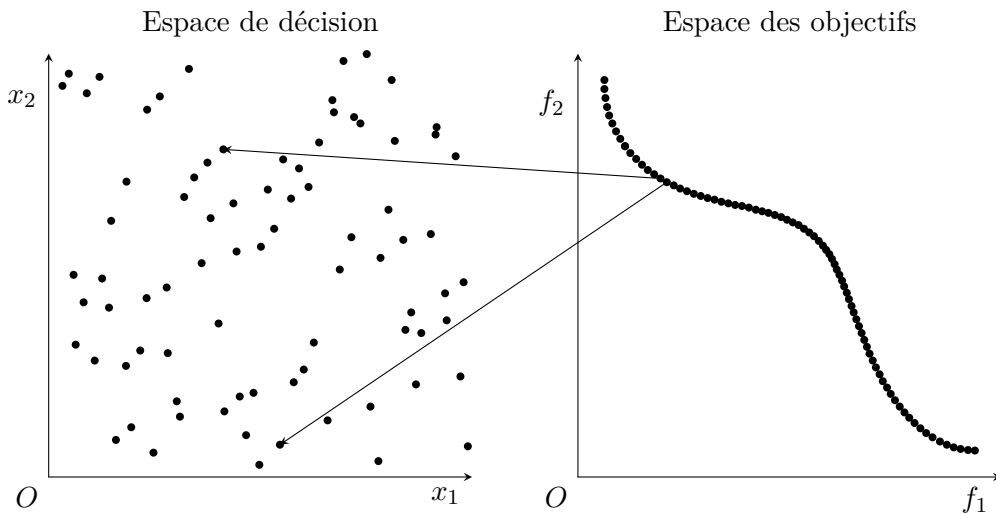


Figure 2.12 – Exemple de scénario où deux points adjacents sur le front de Pareto sont éloignés dans l'espace de décision.

2.3.2 Algorithmes génétiques utilisés

Cette section présente les algorithmes génétiques utilisés dans la suite de ce mémoire.

2.3.2.1 NSGA-II

L'algorithme *NSGA-II* (*Non Dominated Sorting Genetic Algorithm-II*) est un algorithme génétique mutli-objectif élitiste proposé par Deb *et al.* [DAPM00a, DAPM00b]. Basé sur le rang de Goldberg, *NSGA-II* cherche à obtenir rapidement les solutions efficaces du problème tout en conservant une diversité dans l'espace des objectifs. L'élitisme permet à l'algorithme de ne perdre aucune solution efficace (Figure 2.10). La sélection par tournoi est utilisée pour sélectionner les individus qui se reproduiront. La gestion des contraintes est basée sur le critère de contrainte-dominance, où un indice de violation de contraintes regroupe l'ensemble des valeurs des contraintes non satisfaites. L'utilisation de l'opérateur de sélection par tournoi permet de gérer simplement les solutions réalisables et non réalisables.

2.3.2.2 Omni-Optimizer

Proposé par Deb *et al.* [DT08], l'algorithme *Omni-Optimizer* a été conçu pour résoudre une large variété de problèmes d'optimisation, qui étaient jusqu'à présent traités de manière différente, à savoir les problèmes mono- et multi-objectifs avec un ou plusieurs optimum global.

Basé sur *NSGA-II*, l'algorithme *Omni-Optimizer* inclut de nouveaux mécanismes qui lui permette de mieux converger dans certaines situations. Les différences entre *NSGA-II* et *Omni-Optimizer* pour les problèmes multi-objectifs sont listées ci-après :

- le critère de l' ε -dominance est utilisé pour classer les solutions ;
- l'opérateur de sélection restreinte est utilisé pour choisir les individus qui prendront part dans le tournoi ;
- les critères d'évaluation de distance sur l'espace des variables et des objectifs sont utilisés pour maintenir les diversités génotypique et phénotypique sur les surfaces de compromis. La figure 2.13 illustre les calculs des distances phénotypique et génotypique pour un ensemble de solutions appartenant au même front. Le calcul de la distance phénotypique d'une solution i est basée sur la recherche des solutions les proches pour chaque variable.

Les procédures de l'algorithme *Omni-Optimizer* sont présentées en annexe page 148.

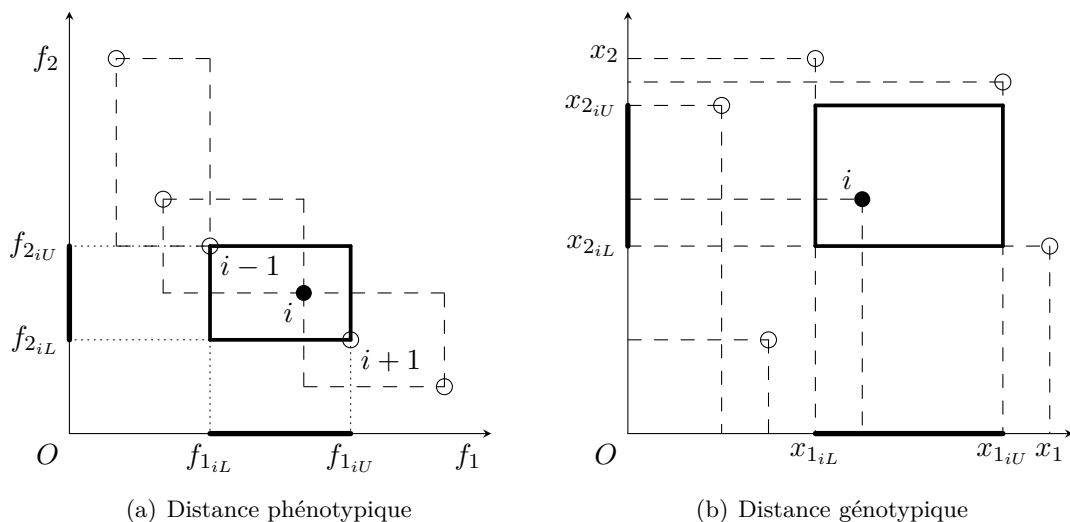


Figure 2.13 – Calculs des distances phénotypique et génotypique pour un problème bi-objectif à deux variables. Image empruntée à Deb *et al.* [DT08].

2.3.3 Discussion

Au fur et à mesure de leurs développements, les algorithmes génétiques se sont révélés être des outils d'optimisation robustes et efficaces, capables de traiter une large palette de problèmes, regroupant les problèmes continus, discrets, mixtes, combinatoires, mono- et multi- objectifs. Ils permettent une exploration efficace de l'espace de recherche et génèrent un historique de l'exploration utile pour le choix final du concepteur. Les développements actuels se penchent sur le couplage d'algorithmes d'optimisation exploitant les forces de chacune des méthodes : les algorithmes génétiques avec méthodes de gradients [DG01] ainsi que les algorithmes génétiques associés à des méta-modèles sont quelques-uns des exemples de couplage d'algorithmes. La richesse des opérateurs génétiques proposés permet d'exploiter au mieux les informations du problème et générer une large palette de solutions efficaces permettant à l'utilisateur d'effectuer un choix a posteriori. La tendance actuelle s'oriente aussi vers les micro-algorithmes génétiques, utilisant un nombre très petit nombre d'individus permettant ainsi un faible nombre d'évaluations des fonctions objectifs. *AMGA* (*Archive-based Micro Genetic Algorithm*) est un des nombreux exemples de micro-algorithmes génétiques développés récemment [TFKD08].

2.4 Évaluation de la qualité des surfaces de compromis

Cette section présente différents indicateurs multi-objectifs, qui seront utilisés pour comparer les résultats obtenus dans les chapitres 3 et 5.

Les algorithmes d'optimisation multi-objectif produisent un ensemble de solutions qui approxime la frontière de Pareto. Cet ensemble de points est appelé surface de compromis. Idéalement, les surfaces de compromis sont confondues avec le front de Pareto.

D'une optimisation à l'autre, les algorithmes stochastiques ne produisent pas les mêmes ensembles de solutions. La comparaison de surfaces de compromis permet de valider le comportement des algorithmes d'optimisation. Pour comparer des ensembles de solutions efficaces, plusieurs métriques ou mesures ont été développées. Chaque métrique a pour objectif de fournir une mesure numérique caractéristique de la surface de compromis. Parmi ces caractéristiques, la proximité par rapport au front de Pareto, la qualité d'approximation du front de Pareto, la diversité des solutions occupent une place importante.

Deux types de métriques sont utilisés : les métriques unaires ou absolues, qui analysent une surface de compromis ou la comparent au front de Pareto. Cette dernière éventualité est possible si le front de Pareto est connu ou tout au moins une approximation de celui-ci. Les métriques binaires ou relatives permettent de comparer deux surfaces de compromis. L'ensemble des indicateurs présentés ici sont des mesures effectuées sur l'espace des objectifs.

Soit un problème multi-objectif avec p objectifs. Soit Q l'ensemble des solutions de la surface de compromis. On note $|Q|$ le cardinal de l'ensemble des points de la surface de compromis. Soit P^* l'ensemble des solutions du front de Pareto, ou du moins l'ensemble approximant le front de Pareto. Les différents éléments se référant au front de Pareto se verront affublés de l'exposant $*$.

Par la suite, les indices i , j et k se référeront respectivement aux variables, objectifs et individus. L'indice l sera utilisé pour parcourir les individus d'une surface de compromis ou du front de Pareto. On appelle \mathbf{x}_k le vecteur de décision à n -dimensions de l'individu k et \mathbf{f}_k le vecteur des fonctions objectifs à p -dimensions de l'individu k .

$$\begin{aligned}\mathbf{x}_k &= (x_{k,1}, \dots, x_{k,i}, \dots, x_{k,n}) \\ \mathbf{f}_k &= (f_{k,1}, \dots, f_{k,j}, \dots, f_{k,p})\end{aligned}\tag{2.8}$$

2.4.1 Indicateurs unaires

Les indicateurs unaires permettent d'analyser une surface de compromis ou encore de la comparer avec le front de Pareto, quand celui-ci est connu.

2.4.1.1 Mesure de la distance entre la surface de compromis et le front de Pareto

Cette métrique permet d'évaluer la distance moyenne entre une surface de compromis et le front de Pareto du problème.

$$\mathcal{D} = \frac{1}{|Q|} \sum_{k=1}^{|Q|} d_k \quad \text{avec} \quad d_k = \min_{l \in \{1, \dots, |P^*|\}} \sqrt{\sum_{j=1}^p (f_{k,j} - f_{l,j}^*)^2} \quad (2.9)$$

d_k correspond à la distance euclidienne dans l'espace des objectifs entre la solution $k \in Q$ et le membre *le plus proche* de P^* . \mathcal{D} permet de mesurer la qualité des points non dominés identifiés. Toutefois, cette mesure ne renseigne pas sur la quantité du front de Pareto découvert. La valeur de \mathcal{D} peut être très faible, signe de bonne qualité de la surface de compromis, mais ne comporter qu'un seul point.

2.4.1.2 Mesure de la représentation du front de Pareto

Cette métrique permet d'évaluer quelle portion du front de Pareto a été découvert par la surface de compromis. Son calcul est très similaire à la mesure précédente.

$$\mathcal{R} = \frac{1}{|P^*|} \sum_{k=1}^{|P^*|} d_k \quad \text{avec} \quad d_k = \min_{l \in \{1, \dots, |Q|\}} \sqrt{\sum_{j=1}^p (f_{l,j} - f_{k,j}^*)^2} \quad (2.10)$$

d_k correspond à la distance euclidienne dans l'espace des objectifs entre la solution $k \in P^*$ du front de Pareto et le membre *le plus proche* de Q . L'idéal est de faire tendre cette mesure vers zéro.

2.4.1.3 Métrique d'espacement \mathcal{S}

Introduite par Schott [Sch95], la métrique d'espacement \mathcal{S} (*Spacing*) permet de caractériser l'uniformité de la répartition des points composant une surface de compromis.

$$\mathcal{S} = \sqrt{\frac{1}{|Q|} \sum_{k=1}^{|Q|} (d_k - \bar{d})^2} \quad (2.11)$$

avec $d_k = \min_{l \in \{1, \dots, |Q|\} \setminus k} \sum_{j=1}^p |f_{l,j} - f_{k,j}|$ et $\bar{d} = \sum_{k=1}^{|Q|} d_k / |Q|$ où \bar{d} est la moyenne de l'ensemble des d_k . La métrique d'espacement \mathcal{S} mesure donc l'écart-type entre les différentes distances d_k . La norme 1 est ici utilisée pour évaluer d_k , on trouve aussi des implémentations où la norme euclidienne est utilisée. De même, le dénominateur n'est pas forcément $|Q|$, mais peut être $|Q| - 1$.

Cette métrique est intéressante pour connaître la distribution des points le long d'une surface de compromis. Toutefois, son résultat peut être biaisé dans le cas, où la surface de compromis est discontinue.

2.4.1.4 Mesure de diversité phénotypique

La métrique précédente présente un autre inconvénient. Elle ne permet pas d'évaluer la quantité du front de Pareto approximé : les solutions peuvent très bien présenter une bonne distribution sur la surface de compromis mais ne pas représenter l'ensemble du front de Pareto. Deb *et al.* [DAPM00b] ont proposé de modifier cette métrique en intégrant les distances avec les solutions extrémales du front de Pareto. Cette nouvelle mesure de diversité s'exprime mathématiquement de la manière suivante :

$$\Delta = \frac{\sum_{j=1}^p d_j^e + \sum_{k=1}^{|Q|} |d_k - \bar{d}|}{\sum_{j=1}^p d_j^e + |Q| \bar{d}} \quad (2.12)$$

avec d_j^e les distances entre les points extrémaux de la surface de compromis et ceux du front de Pareto. Les termes d_k et \bar{d} sont identiques à ceux de la métrique d'espacement \mathcal{S} .

2.4.1.5 Mesure de la distance phénotypique maximale

Introduite par Zitzler [Zit99], cette mesure évalue la longueur de la diagonale de l'hyperboîte définie par les points extrêmes identifiés lors d'une simulation.

$$D = \sqrt{\sum_{j=1}^p \left(\max_{k \in \{1, \dots, |Q|\}} f_{k,j} - \min_{k \in \{1, \dots, |Q|\}} f_{k,j} \right)^2} \quad (2.13)$$

Pour un problème bi-objectif, elle est équivalente à la distance entre les deux solutions extrémales. La figure 2.14 présente cet indicateur sur un problème de minimisation bi-objectif. Cet indicateur a par la suite été normalisé à l'aide des points extrêmes du front du Pareto, de manière à obtenir des valeurs comprises dans l'intervalle $[0; 1]$

$$\bar{D} = \sqrt{\frac{1}{p} \sum_{j=1}^p \left(\frac{\max_{k \in \{1, \dots, |Q|\}} f_{k,j} - \min_{k \in \{1, \dots, |Q|\}} f_{k,j}}{\max_{k \in \{1, \dots, |P^*|\}} f_{k,j}^* - \min_{k \in \{1, \dots, |P^*|\}} f_{k,j}^*} \right)^2} \quad (2.14)$$

Cette mesure nécessite donc de connaître le front de Pareto du problème, ou du moins une meilleure estimation que la surface de compromis testée. D et \bar{D} ne permettent pas de connaître la répartition des solutions sur le front de Pareto.

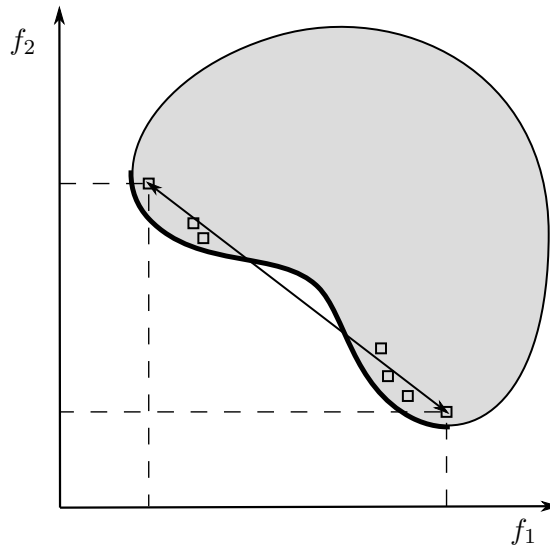


Figure 2.14 – Calcul de la distance phénotypique maximale. Cette mesure ne rend pas compte de la distribution des points le long de la surface de compromis.

2.4.1.6 Mesure de l'hypervolume

Proposée par Zitzler [ZDT00], la mesure de l'hypervolume, aussi connue sous le nom de mesure de Lebesgue, évalue le volume couvert par les solutions de Q dans l'espace des objectifs. Pour chaque solution $i \in Q$, un hypercube v_i est construit à partir d'un point de référence $\mathbf{W} \in \mathbb{R}^p$, la solution i se trouvant à l'extrémité de la diagonale de v_i et partant de \mathbf{W} . L'hypervolume est ensuite calculé comme le volume de l'union des différents hypercubes :

$$HV(Q) = \text{volume} \left(\bigcup_{i=1}^{|Q|} v_i \right) \quad (2.15)$$

La figure 2.15 illustre le calcul de l'hypervolume pour une série de points non dominés dans un problème de minimisation bi-objectif. Le cercle noir représente le point Nadir et le carré noir le point de référence \mathbf{W} . Idéalement, on choisirait le point Nadir comme point de référence, toutefois dans la pratique, ce point n'est pas connu et sa détermination est compliquée pour les problèmes avec

plus de deux objectifs. On choisit donc un point de référence \mathbf{W} dominé par l'ensemble des points non dominés. Plus grand est l'hypervolume, meilleure est la qualité des points non dominés obtenus. L'évolution de l'hypervolume donne une bonne information sur l'évolution de la convergence de la population.

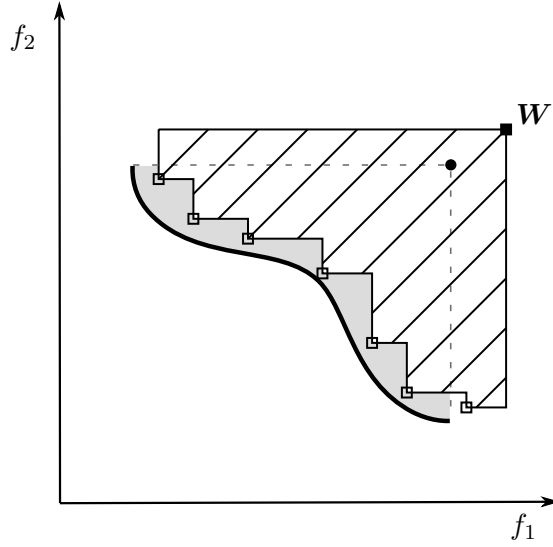


Figure 2.15 – Calcul de l'hypervolume pour un problème de minimisation bi-objectif à partir des points non-dominés trouvés.

Si l'on connaît le front de Pareto ou une approximation, cette métrique peut être normée pour rendre la comparaison des résultats plus aisée [Van99].

$$HVR(Q) = \frac{HV(Q)}{HV(P^*)} \quad (2.16)$$

Les résultats de cette métrique sont compris entre zéro et un. Enfin, il est intéressant de noter que des algorithmes génétiques ont basé leur opérateur de sélection sur la valeur de l'hypervolume (IBEA [ZK04], HypE [BZ08]).

Les différents éléments géométriques nécessaires au calcul des métriques pour un problème de minimisation bi-objectif sont résumés sur la figure 2.16.

2.4.2 Indicateurs binaires

Les indicateurs binaires permettent de comparer deux surfaces de compromis entre elles. Ces indicateurs sont utilisés pour comparer les performances de différents algorithmes évolutionnaires. La mesure la plus utilisée est la métrique de couverture d'ensemble, appelée métrique \mathcal{C} .

2.4.2.1 Métrique \mathcal{C}

Introduite par Zitzler *et al.* [ZDT00], la métrique \mathcal{C} permet de comparer deux surfaces de compromis A et B . Elle évalue la proportion de solutions de B qui sont dominées par les solutions de A . Elle se calcule de la manière suivante

$$\mathcal{C}(A, B) = \frac{|\{b \in B \mid \exists a \in A : a \preceq b\}|}{|B|} \quad (2.17)$$

Cette métrique est comprise entre dans l'intervalle $[0; 1]$. Lorsque $\mathcal{C}(A, B) = 1$, toutes les solutions de B sont dominées ou égales aux solutions de A . À l'inverse, lorsque $\mathcal{C}(A, B) = 0$, aucune des solutions de B n'est dominée par les solutions de A . Comme l'opérateur de dominance n'est pas un opérateur symétrique, $\mathcal{C}(A, B)$ n'est pas nécessairement égale à $1 - \mathcal{C}(B, A)$. La comparaison de deux surfaces de compromis conduit donc à deux valeurs pour la métrique \mathcal{C} .

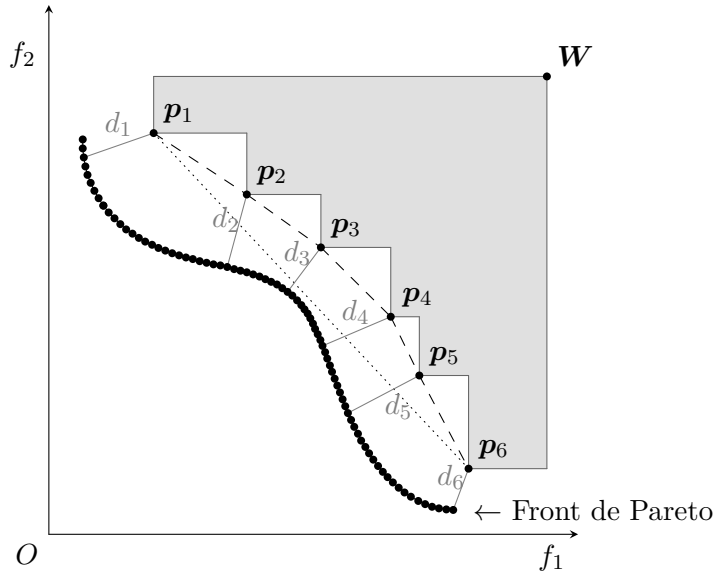


Figure 2.16 – Présentation des différents éléments géométriques nécessaires au calcul des métriques pour un problème de minimisation bi-objectif. Les points $p_i, \forall i \in \{1, \dots, 6\}$ forme une surface de compromis. Les distances $d_i, \forall i \in \{1, \dots, 6\}$ correspondent aux distances minimales entre les points p_i et le front de Pareto et permettent d'évaluer la distance au front de Pareto. Elles sont aussi utilisées pour évaluer la représentation du front de Pareto. Les segments en traits discontinus sont utilisés pour évaluer la diversité de la surface de compromis. La longueur du segment en pointillé donne la distance phénotypique maximale. Enfin, l'aire du domaine grisé correspond à l'hypervolume calculé avec le point W .

2.4.3 Discussion

Chacune de ces métriques exprime un aspect (forme, qualité, diversité, ...) de la surface de compromis. Plusieurs d'entre elles sont nécessaires pour se faire une idée de la surface de compromis. Enfin pour les problèmes bi- et tri-objectifs, le dessin des surfaces de compromis fournit de nombreuses indications.

L'ensemble des mesures présentées est réalisé dans l'espace des objectifs. Certaines mesures peuvent aussi être calculées dans l'espace des variables, notamment les différentes mesures de diversité. Pas ou peu utilisée dans la littérature, la mesure de la diversité génotypique permet de savoir si la surface de compromis a été obtenue avec un ensemble de solutions semblables ou pas. Le choix d'un algorithme évolutionnaire multi-objectif s'est longtemps fait sur sa capacité à proposer à des surfaces de compromis, où les points sont bien répartis le long du front de Pareto. Toutefois, dans de nombreux problèmes notamment industriels, le choix d'une solution finale se fait en fonction des objectifs, mais aussi en fonction des variables. Il est donc fort intéressant de comparer les algorithmes suivant ce critère. La mesure de la diversité génotypique est plus compliquée à établir que la diversité phénotypique, car les problèmes d'optimisation utilisent, dans bien des cas, différents types de variables. Par conséquent, il n'existe pas de mesure unique pour caractériser cette diversité.

2.5 Conclusion

Les problèmes de placement sont dans la majorité des cas des problèmes multi-objectifs, pour lesquels des méthodes d'optimisation robustes sont nécessaires. L'analyse de l'état de l'art a montré que l'ensemble des techniques d'optimisation a été utilisé pour leurs résolutions.

Les algorithmes évolutionnaires, plus particulièrement les algorithmes génétiques, ont suscité un fort intérêt pour la résolution des problèmes de placement. Ces algorithmes stochastiques ont nécessité le développement de métriques pour pouvoir analyser et comparer les résultats proposés.

L'exemple de problème d'agencement traité dans le chapitre suivant utilise un algorithme génétique combinatoire pour trouver un ensemble de solutions efficaces. Enfin, la méthode proposée chapitre 4 utilise un algorithme génétique pour effectuer une optimisation globale.

Chapitre 3

Optimisation multi-objectif d'agencement de locaux

Ce chapitre présente une méthode d'agencement, où la géométrie des composants n'intervient pas directement. La méthode est appliquée sur un exemple d'agencement de compartiments d'un navire. Modélisé comme un problème d'optimisation combinatoire multi-objectif, la résolution repose sur une méthode de placement originale garantissant le respect des contraintes de placement.

Sommaire

3.1	Introduction	44
3.2	Objectifs du problème	44
3.3	Étapes de la modélisation	45
3.3.1	Données du problème	45
3.3.2	Codage de l'information	46
3.3.3	Correspondance entre codage et agencement	47
3.3.4	Évaluation des individus	47
3.3.5	Calcul des distances entre compartiments	49
3.4	Optimisation multi-objectif	50
3.4.1	Variables d'optimisation	50
3.4.2	Choix de l'algorithme d'optimisation	50
3.4.3	Surfaces de compromis	50
3.4.4	Analyse des résultats	53
3.4.5	Extensions futures	54
3.5	Conclusion	57

3.1 Introduction

Pour s'assurer du respect des contraintes de placement, de nombreux problèmes d'agencement utilisent une heuristique de placement ou un schéma d'encodage. Pour étudier plus précisément cette catégorie de problèmes, on choisit d'en résoudre un, à savoir un problème d'agencement de compartiments de navire. Cet exemple est inspiré des travaux de Lee *et al.* pour lesquels une méthode de résolution mono-objectif a été proposée [LHR02, LHR03, LHR05]. Nous proposons ici de généraliser la méthode mono-objectif proposée au cas multi-objectif.

Les problèmes d'agencement de compartiments ou de locaux ont été étudiés depuis de nombreuses années. Deux méthodes ont été élaborées pour traiter les problèmes d'agencement de locaux de différentes tailles : soit l'espace est traité de manière discrète, soit de manière continue. La méthode discrète décompose l'espace alloué en unités élémentaires, les compartiments sont alors placés sur ces unités d'aires. La majorité des modélisations d'agencement de locaux est basée sur ce principe. De nombreux algorithmes reposant sur cette approche ont été développés depuis une cinquantaine d'années : *CRAFT* [AB63], *ALDEP* [SE67], *CORELAP* [LM67], *SPIRAL* [Goe92] et *MULTIPLE* [BME94]. Ces méthodes sont bien adaptées lorsque les composants sont de géométries orthogonales.

Lee *et al.* proposent d'utiliser une représentation continue de l'espace capable de prendre en compte les éléments géométriques particuliers d'un problème, en particulier la forme non orthogonale du contour (navire). Ici, ces éléments correspondent aux structures intérieures du bateau (coque, cloison, escalier, ...). Dans ce problème, la géométrie des compartiments importe peu, par contre on souhaite que chaque compartiment ait une surface au sol minimale. La représentation continue de l'espace est donc bien adaptée à ce problème.

3.2 Objectifs du problème

Les objectifs sont de répartir l'ensemble des compartiments sur les ponts d'un navire afin d'optimiser les critères définis par le concepteur. Le problème est résolu avec quelques contraintes simples :

- l'ensemble des compartiments doit être placé à l'intérieur du navire ;
- les compartiments ne doivent ni chevaucher les couloirs ni les cloisons de structure ;
- l'espace alloué à chaque compartiment doit être au moins supérieur à l'aire requise pour chaque compartiment.

Une méthode de placement séquentiel, qui positionnera les compartiments un à un, sera utilisée pour s'assurer que ces contraintes soient respectées. L'agencement des compartiments répond principalement à deux critères : le premier consiste à minimiser les flux entre compartiments et le second consiste à maximiser une fonction traduisant la volonté de proximité entre certains compartiments exprimée par le concepteur (affinités entre compartiments). Ce second objectif sera par la suite transformé en une minimisation pour reprendre les notations de Lee *et al.* . Mathématiquement, ces deux derniers objectifs reviennent à minimiser les fonctions f_1 et f_2 :

$$\begin{aligned} f_1 &= \sum_{i=1}^{m-1} \sum_{j=i+1}^m (f_{ij} \times d_{ij}) \\ f_2 &= \sum_{i=1}^{m-1} \sum_{j=i+1}^m (C_a - b_{ij} \times c_{ij}) \end{aligned} \tag{3.1}$$

avec :

- m le nombre de compartiments à placer.
- f_{ij} le flux de matériel entre les compartiments i et j . Valeur définie par le concepteur.
- d_{ij} la distance entre les compartiments i et j .
- C_a la valeur maximale d'adjacence entre les compartiments. $C_a = 5$
- c_{ij} le coefficient d'adjacence entre les compartiments i et j . Valeur définie par le concepteur.
- b_{ij} le facteur d'adjacence entre les compartiments i et j . Ce facteur est fonction de la distance d_{ij} : $b_{ij} = 1 - \frac{1}{6} \left\lfloor 6 \frac{d_{ij}}{d_{\max}} \right\rfloor$, avec $d_{\max} = \max \{d_{ij}\}$, $0 \leq b_{ij} \leq 1$

Les données du problème sont les matrices de flux \mathbf{F} et d'adjacence \mathbf{C} , regroupant respectivement les termes f_{ij} et c_{ij} . Les valeurs des flux sont normalisées entre 0 et 1. Plus ces valeurs sont proches de 1, plus les compartiments associés devraient être proche l'un de l'autre. Si le flux f_{ij} est nul, alors la position relative du compartiment j par rapport au compartiment i n'influence pas la fonction f_1 . Les valeurs de la matrice d'adjacence \mathbf{C} , qui expriment le souhait de proximité entre chaque paire de compartiments, sont comprises entre 0 et C_a : $0 \leq c_{ij} \leq C_a$. Les matrices \mathbf{D} et \mathbf{B} sont fonction de l'agencement choisi, elles doivent être calculées pour chaque nouvelle répartition des compartiments. La matrice \mathbf{D} regroupe l'ensemble des distances entre compartiments. La matrice \mathbf{B} contient les facteurs d'adjacence d'un agencement. Elle permet d'identifier les paires de compartiments proches ou éloignés avec une mesure comprise entre 0 et 1. Pour éviter les calculs inutiles, seuls les termes d_{ij} , pour lesquels f_{ij} ou c_{ij} ne sont pas nuls, sont calculés. De toute évidence, les matrices \mathbf{D} , \mathbf{B} et \mathbf{C} sont symétriques, la matrice de flux \mathbf{F} peut quant à elle ne pas être symétrique : le flux f_{ij} peut être différent du flux f_{ji} . Par la suite, la matrice de flux \mathbf{F} est considérée symétrique.

Lee *et al.* ont résolu ce problème dans le cadre d'une optimisation mono-objectif, où le problème est ramené à minimiser une seule fonction F .

$$F = \omega_1 \sum_{i=1}^{m-1} \sum_{j=i+1}^m (f_{ij} \times d_{ij}) + \omega_2 \sum_{i=1}^{m-1} \sum_{j=i+1}^m (C_a - b_{ij} \times c_{ij}) \quad (3.2)$$

où ω_1 et ω_2 servent à pondérer les deux objectifs. Comme mentionné précédemment, cette optimisation permet d'obtenir uniquement les points supportés de la frontière de Pareto, et plusieurs optimisations sont nécessaires pour obtenir un ensemble de solutions. Dans le cadre d'une optimisation multi-objectif, il n'y a pas d'agrégation des objectifs, et les solutions ou individus sont classés selon la méthode des rangs de Goldberg (*c.f.* section 2.2).

3.3 Étapes de la modélisation

Cette section présente les différentes étapes permettant de passer des variables d'optimisation aux valeurs des objectifs.

3.3.1 Données du problème

La première étape consiste à fixer une géométrie des ponts, et calculer les aires disponibles sur chacun de ces ponts. Les figures 3.1 et 3.2 présentent les différents ponts d'un navire simplifié. Sur ces figures sont représentées la coque du navire et les structures internes à savoir les cloisons et les couloirs. Dans la colonne de droite de la figure 3.2, sont représentées les surfaces disponibles sur chaque pont en fonction de la position x . Par la suite, les surfaces des ponts seront divisées en fonction des cloisons et des couloirs. Le contour de chaque pont, limité par la coque du navire, est défini de manière discrète, c'est-à-dire que le modèle actuel est adaptable à tout type de coque.

Dans l'exemple que l'on résout, on fixe le nombre de compartiments à 78 et on leur attribue une aire minimale définie aléatoirement entre deux bornes. La compacité du problème exprimant le rapport entre l'ensemble des aires des compartiments et l'espace disponible est de 85%. Les 15% restants seront utilisés pour éventuellement agrandir les surfaces des 78 compartiments.

Une fois que la géométrie du navire, le nombre de compartiments ainsi que leurs surfaces nécessaires ont été fixés, les données sur les compartiments à placer sont créées. Cela revient à transcrire les données utilisateurs dans des matrices résumant leurs souhaits. Il s'agit de la matrice flux \mathbf{F} et de la matrice d'adjacence \mathbf{C} . Dans notre cas, les flux entre les compartiments sont générés aléatoirement avec une probabilité fixée par l'utilisateur (20%). La matrice \mathbf{C} contenant les affinités d'agencement est déterminée à partir de la matrice de flux et d'une perturbation, de manière à ce que les données ne soient pas complètement contradictoires. La figure 3.3 résume l'ensemble des données du problème d'agencement.

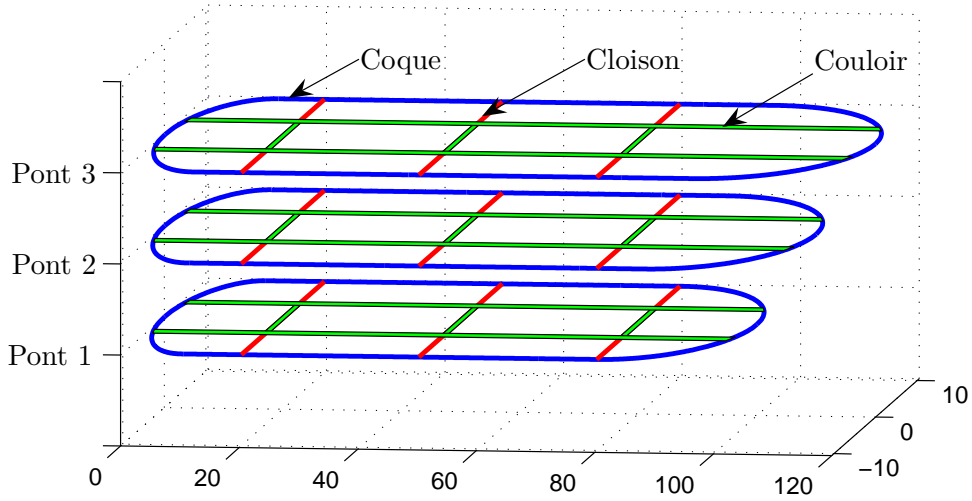


Figure 3.1 – Vue 3D des différents ponts de l'exemple considéré.

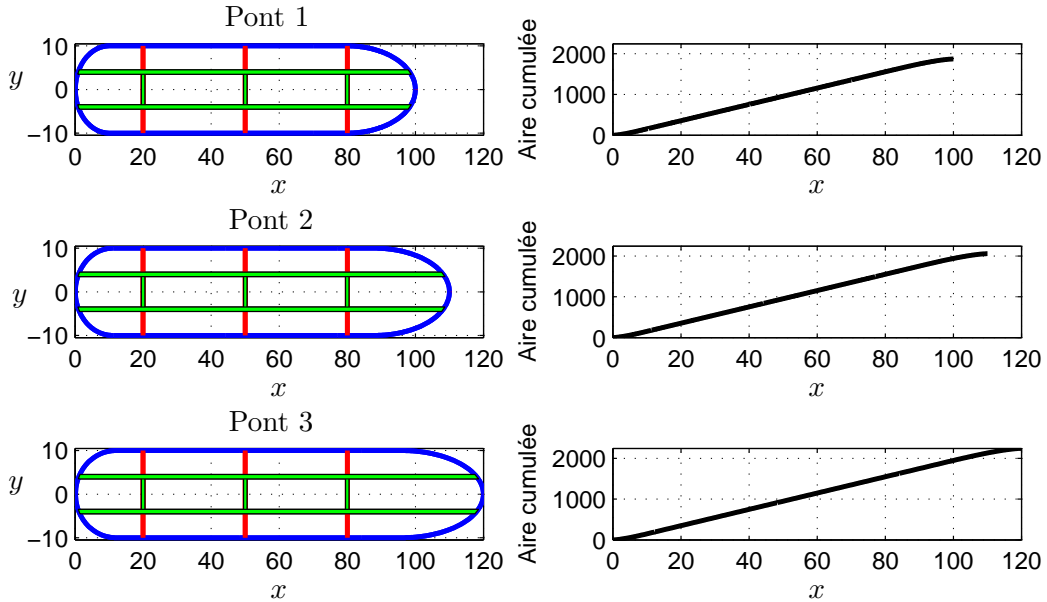


Figure 3.2 – Vue 2D des différents ponts de l'exemple considéré. L'évolution de l'aire cumulée de chaque pont est aussi donnée.

3.3.2 Codage de l'information

Le codage permet d'établir une convention décrivant chaque solution possible sous la forme d'une chaîne de caractères. Les informations relatives à une solution sont codées sur des chromosomes. Ces chromosomes sont composés de deux parties. La première donne l'ordre d'introduction des compartiments sur les ponts et leur aire associée, la seconde renseigne sur la position des couloirs. Les opérations génétiques effectuées sur le chromosome seront différentes selon la partie considérée. Le premier segment comporte m éléments, qui correspondent à la permutation des compartiments, le second en comporte autant et correspond à l'aire associée à chaque compartiment. La seconde partie contient les positions horizontales et verticales des couloirs de chaque pont. Ces positions sont représentées à l'aide d'entiers. Les positions des ascenseurs et escaliers, permettant la communication entre les différents

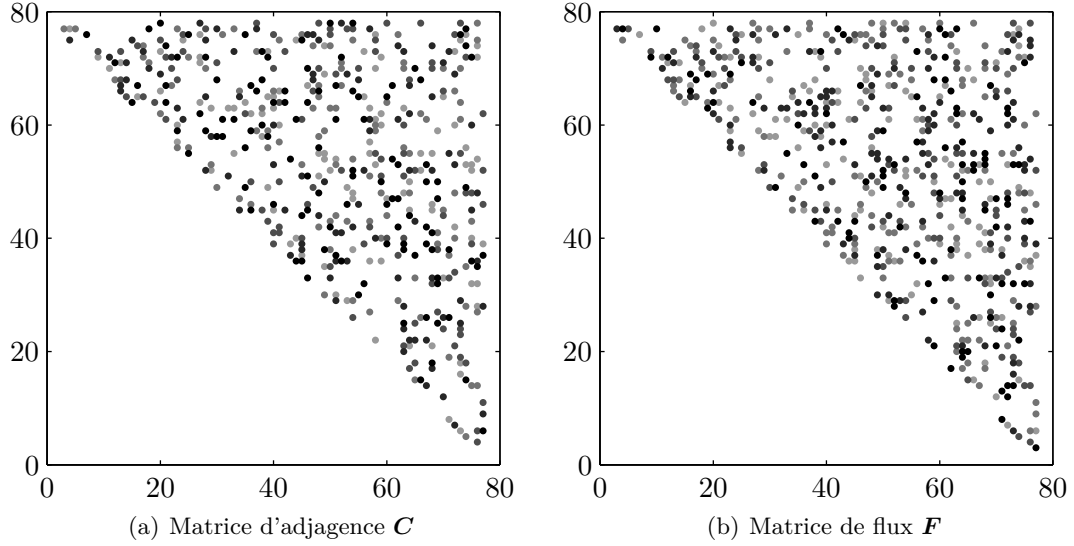


Figure 3.3 – Densité des matrices de données d'adjacence et de flux pour les couples de compartiments du navire. Les niveaux de gris renseignent sur l'intensité des connexions entre chaque couple de compartiments.

ponts, sont réparties de manière égale le long de chaque couloir. La table 3.1 présente le codage d'une solution.

Pont	1 ^{ère} partie		2 ^{de} partie	
	Ordre des compartiments	Aire associée à chaque compartiment	Positions couloirs horizontaux	Positions couloirs verticaux
Nombre d'éléments	m	m	$n_p(n_h + 1)$	$n_p(n_v + 1)$

Table 3.1 – Représentation des données pour le problème d'agencement de navire. (n_p = Nombre de ponts, n_h et n_v = Nombres de couloirs horizontaux et verticaux par pont).

3.3.3 Correspondance entre codage et agencement

La figure 3.4 illustre le décodage d'un chromosome. La première étape consiste à positionner les couloirs sur les différents ponts. Par la suite, il faut évaluer les fonctions d'aires cumulées pour chaque sous-zone. Les compartiments peuvent ensuite être positionnés. L'heuristique de placement positionne les compartiments les uns à la suite des autres dans les différentes sous-zones. Cette répartition des compartiments se fait en prenant en compte la géométrie et les obstacles des ponts. Si l'aire disponible de la sous-zone est inférieure à l'aire du compartiment à positionner, il est placé dans la suivante. Les flèches en trait semi-continu de la figure 3.4 indiquent le sens de balayage des ponts pour le positionnement des compartiments. Cette méthode de placement génère des espaces vides, qui permettront l'agrandissement de certains compartiments. À la fin, il se peut que l'ensemble des compartiments n'ait pas pu être placé. Dans ce cas là, la solution est irréalisable et la somme des aires des composants qui n'ont pas pu être placés, est utilisée comme une pénalité dans l'algorithme d'optimisation.

3.3.4 Évaluation des individus

Pour chaque solution, l'heuristique de placement répartit les compartiments sur les différents ponts et calcule les positions de centre de gravité de chaque compartiment. Ces positions sont utilisées pour le calcul de distances entre compartiments. Une fois ces distances calculées, les objectifs peuvent ensuite être évalués. Les étapes de la modélisation sont résumées ici :

1. Décodage d'un chromosome ;
2. Répartition géométrique des compartiments ;

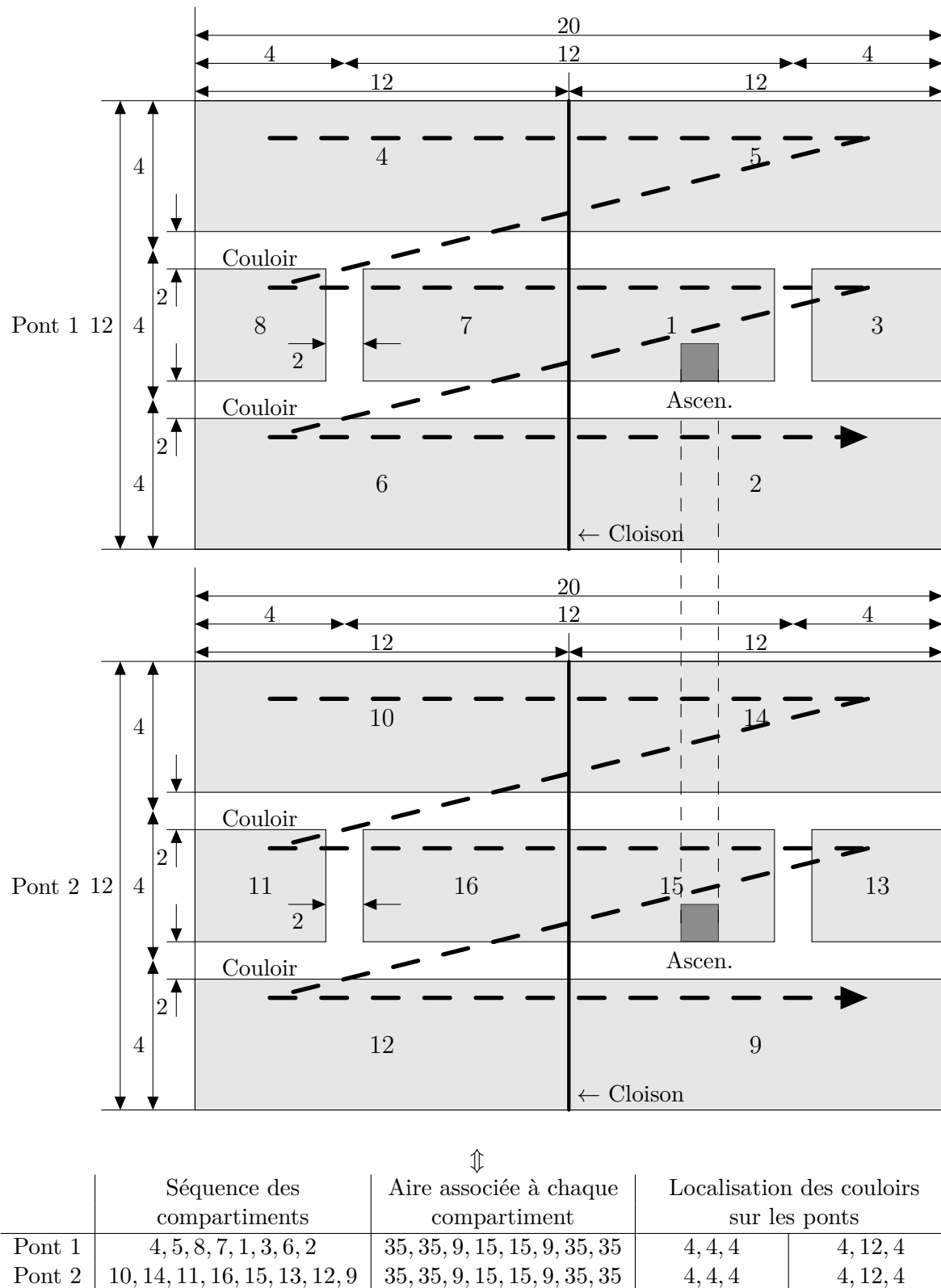


Figure 3.4 – Encodage / Décodage des données pour le problème d'agencement de compartiments d'un navire. Le terme *Ascen.* est une abréviation pour ascenseur.

3. Calcul des positions de centre de gravité des compartiments ;
4. Calculs des distances entre compartiments ;
5. Évaluation des fonctions f_1 et f_2 .

3.3.5 Calcul des distances entre compartiments

Deux méthodes sont envisageables pour évaluer les distances entre compartiments. Une première méthode consiste à calculer ces distances directement avec la distance de Manhattan. Pour cela, il suffit de connaître les coordonnées des deux compartiments, le calcul est immédiat. Cette méthode est approchée, mais fournit des résultats extrêmement rapidement. Dans ce cas là, la distance entre deux compartiments C_i et C_j s'écrit :

$$d_{ij} = |x_i - x_j| + |y_i - y_j| + |z_i - z_j| \quad (3.3)$$

où (x_i, y_i, z_i) et (x_j, y_j, z_j) désignent respectivement les coordonnées des centres de gravité des compartiments C_i et C_j .

La deuxième méthode, qui sera utilisée par la suite, consiste à calculer les distances exactes entre les compartiments. Pour cela, il faut calculer les longueurs des chemins les plus courts reliant chaque paire de compartiments en interaction. La théorie des graphes et l'algorithme de Dijkstra [Dij59] sont utilisés pour résoudre ce problème. Il s'agit de créer l'ensemble des chemins possibles entre deux compartiments et de trouver le plus court. Concrètement, pour chaque solution, on liste l'ensemble des points de passage dans les couloirs du navire. Typiquement, un point de passage est créé devant chaque compartiment. D'autres points sont créés au niveau des intersections de couloir et devant chaque escalier et ascenseur. Ces points de passage sont appelés des nœuds. Ensuite, les connexions définissant les chemins de longueur minimale entre les différents nœuds sont établies. Plusieurs règles sont utilisées pour créer ces connexions. L'ensemble des nœuds d'un même couloir sont reliés entre eux ainsi qu'aux nœuds donnant accès au couloir et ascenseur de ce couloir. Des connexions sont aussi créées entre les ponts via les escaliers et ascenseurs. D'autres règles peuvent être ajoutées en fonction de chaque navire. Les connexions permettent de définir l'ensemble des chemins à bord du navire. L'ensemble de ces résultats est représenté sous la forme d'une matrice triangulaire supérieure M , où M_{ij} donne la distance entre les nœuds i et j . Enfin, l'algorithme de Dijkstra est utilisé pour trouver le chemin le plus court entre deux compartiments à partir de la matrice de distance M . De complexité polynomiale, cet algorithme exact renvoie la liste des nœuds à emprunter pour relier deux compartiments ainsi que la distance les séparant.

La figure 3.5 illustre cette différence de calcul de distance. Le calcul des vraies distances multiplie les temps de calcul par trois par rapport à un simple calcul de la distance de Manhattan. Les différences entre ces distances pouvant être importantes, les optimisations réalisées dans la section suivante calculeront les distances exactes entre les compartiments.

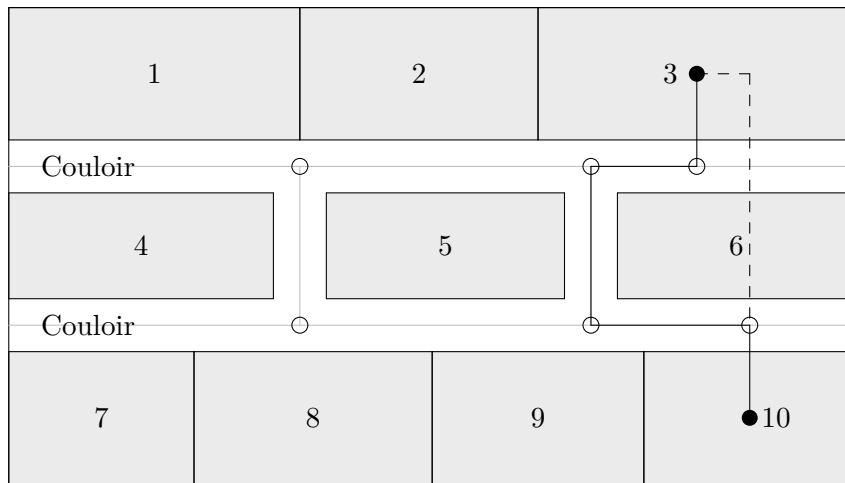


Figure 3.5 – Différence entre calculs approché et réel des distances entre compartiment. En pointillé est représentée la distance de Manhattan qui approxime la distance entre les compartiments 3 et 10. En trait plein est représentée la distance réelle entre ces deux compartiments.

3.4 Optimisation multi-objectif

Les différentes optimisations réalisées pour résoudre ce problème d'agencement sont présentées dans cette section.

3.4.1 Variables d'optimisation

Par la suite, la position des couloirs est considérée fixe. On effectue ce choix de manière à analyser le comportement de l'algorithme d'optimisation. Cela signifie que les calculs des fonctions d'aire pour chaque sous-zone peuvent être effectués une seule fois. La variable d'optimisation est donc uniquement la permutation qui gère l'ordre d'introduction des compartiments sur les différents ponts. Le nombre de permutations est fini, toutefois le nombre de solutions reste très élevé : ce nombre est égal à la factorielle du nombre de compartiments. Avec 78 compartiments, le nombre de solutions est égal à 1.1324×10^{115} .

Devant le nombre exponentiel de solutions possibles, il devient évident qu'une approche déterministe ne permet pas de trouver une solution correcte dans un temps fini. Seule une approche stochastique peut permettre de trouver une solution.

3.4.2 Choix de l'algorithme d'optimisation

Nous avons choisi l'algorithme génétique multi-objectif *NSGA-II* pour effectuer l'optimisation. Les objectifs de l'algorithme consiste donc à minimiser les fonctions f_1 et f_2 présentées auparavant. Les opérateurs génétiques utilisés sont spécifiques aux permutations. L'opérateur de croisement choisi est la version multi-objectif de l'opérateur proposé par Lee *et al.* [LHR03]. Cet opérateur est basé sur le principe que les meilleures solutions doivent transmettre davantage d'informations que les solutions moins bonnes. Dans le cas multi-objectif, on utilise le rang des solutions pour déterminer quelle proportion d'informations de la permutation sera transmise aux enfants. D'autres opérateurs comme les opérateurs *OX* ou *PMX* [Sys91] auraient pu être utilisés. L'opérateur de mutation intervertit au hasard deux compartiments dans la permutation avec une probabilité d'action.

Les paramètres de ces opérateurs sont résumés dans la table 3.2. On peut constater que le nombre de solutions évaluées est au maximum 900×128 soit 115200, nombre extrêmement petit par rapport au nombre de solutions existantes. La forte probabilité de croisement permet de générer de nombreuses solutions différentes à chaque nouvelle génération. La mutation permet d'apporter des modifications locales des solutions avec une probabilité d'action de 10%.

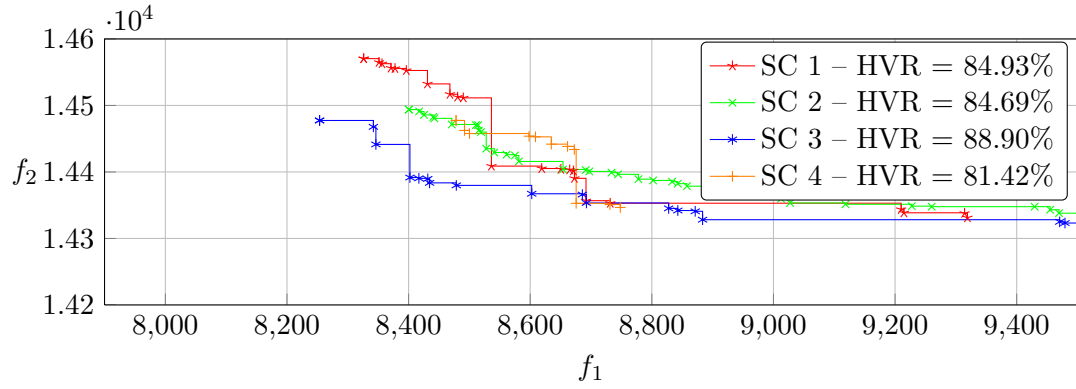
Paramètres	Valeur
Nombre de générations	300 - 600 - 900
Nombre d'individus	128
Probabilité de croisement	0.9
Probabilité de mutation	0.1

Table 3.2 – Paramètres de l'algorithme génétique utilisé

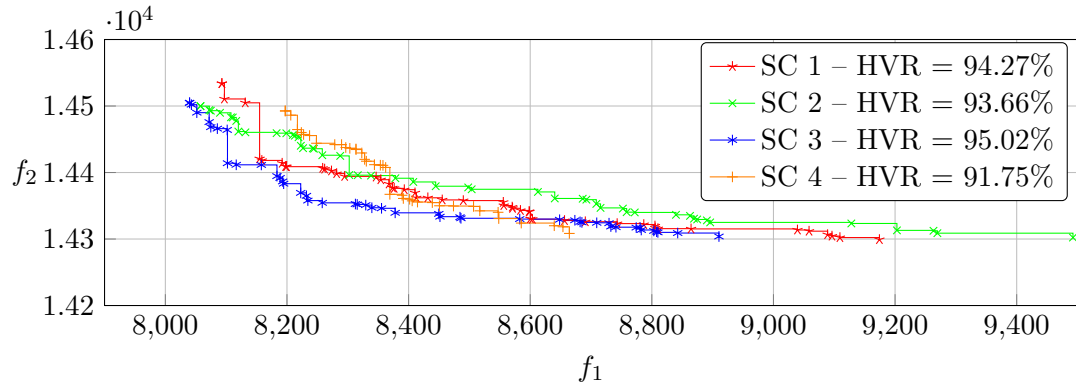
3.4.3 Surfaces de compromis

On se propose de comparer quatre exécutions de l'algorithme génétique. Pour chaque exécution, le germe des séries des nombres aléatoires est différent. Chaque population initiale générée aléatoirement sera donc différente des autres, de même que les nombres aléatoires utilisés pour les différentes opérations génétiques.

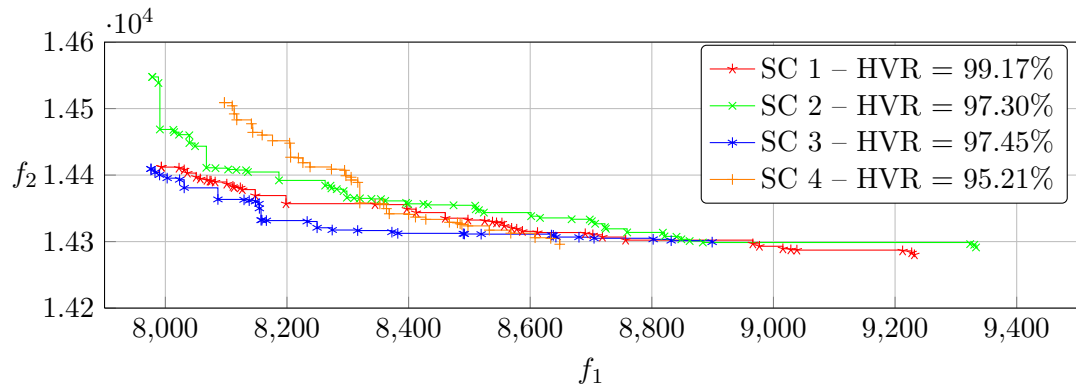
La figure 3.6 présente l'évolution des quatre surfaces de compromis pour trois générations données 300, 600, 900. Initialement, le nombre de générations avait été réglé sur 300. Toutefois après analyse de l'évolution des surfaces de compromis et de l'hypervolume, ce nombre de générations s'est avéré trop petit. La convergence n'était pas atteinte. Un deuxième essai avec 600 générations ne s'est toujours pas révélé concluant. Une augmentation du nombre de générations à 900 a permis d'obtenir une



(a) Génération : 300



(b) Génération : 600



(c) Génération : 900

Figure 3.6 – Évolution des surfaces de compromis pour quatre optimisations avec des populations initiales différentes pour le problème d'agencement de navires. SC = Surface de compromis. HVR = HyperVolume Relatif. Pour chaque surface de compromis, l'hypervolume relatif est donné. Calculé à partir du front de Pareto que l'on a identifié, l'hypervolume total a été évalué à 3.31×10^6 .

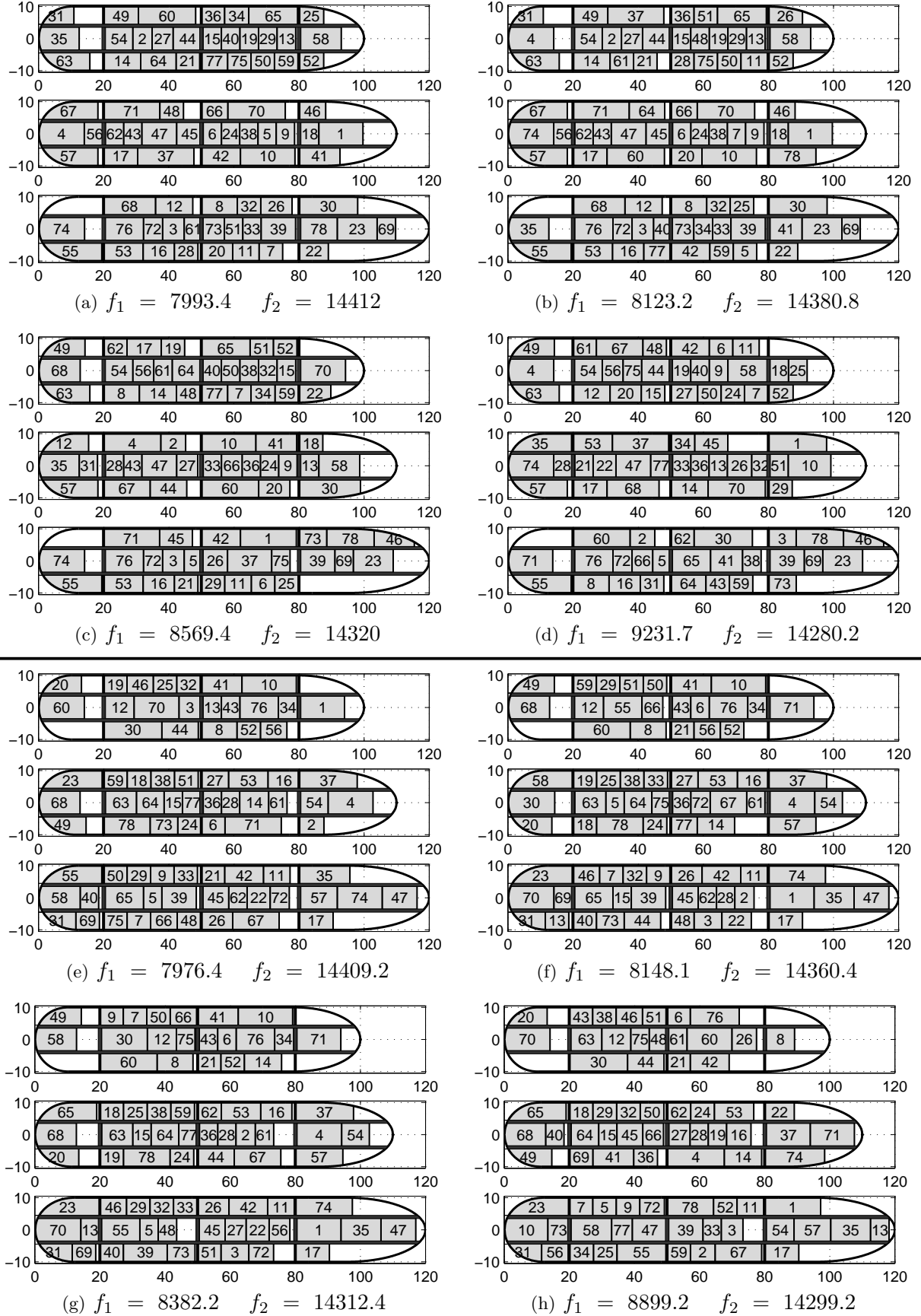


Figure 3.7 – Solutions du problème d'agencement de navire. Les sous-figures (a) à (d) sont issues de la surface de compromis 1 présentée figure 3.6(c). Les sous-figures (e) à (h) sont issues de la surface de compromis 3.

convergence satisfaisante. Un critère d'arrêt basé sur le nombre de générations est donc loin d'être suffisant. D'autres critères d'arrêt comme la stagnation du nombre de nouveaux points ajoutés à la surface de compromis peuvent aussi être utilisés.

La figure 3.7 présente huit solutions pour le problème d'agencement. Les quatre premières sont issues de la surface de compromis de la première optimisation (SC1), les quatre suivantes sont issues de la troisième optimisation (SC3). Les solutions sont triées dans l'ordre croissant du premier objectif. Les solutions (a) et (d) représentent les solutions extrémales de la première optimisation, les solutions (e) et (h) celles de la troisième optimisation. En parcourant les compartiments de gauche à droite et de haut en bas, on retrouve la permutation qui a donné cette solution. On peut observer que les solutions d'une même optimisation sont assez différentes. L'opérateur de diversité phénotypique de *NSGA-II* a donc joué son rôle. Quand on compare les solutions de deux optimisations différentes, les solutions sont encore plus différentes. Cela s'explique par l'absence de contraintes de localisation des compartiments. Par ailleurs sur les solutions présentées, on peut constater qu'il y a de nombreux espaces vides. Ces espaces peuvent être exploités en agrandissant les compartiments voisins. On peut envisager de spécifier une tolérance sur les aires des compartiments, et résoudre le problème pour leurs aires minimales. À la fin de l'optimisation, la taille des compartiments peut être augmentée en fonction de l'espace disponible autour d'eux.

La figure 3.8 présente l'évolution de l'hypervolume de ces quatre optimisations à chaque génération. Présentée section 2.4.1.6, cette métrique évalue l'aire du domaine dominé par les points de la surface de compromis dans l'espace des objectifs. Le point de référence \mathbf{W} utilisé pour évaluer l'hypervolume a été choisi à partir des plus mauvaises valeurs obtenues pour chaque objectif : $\mathbf{W} = (1.34 \times 10^4, 1.49 \times 10^4)$. L'hypervolume de référence calculé à partir des différentes surfaces de compromis identifiées, est évalué à 3.31×10^6 . Chacune de ces courbes est croissante, résultat du choix d'un algorithme génétique élitiste. Avec le choix de l'algorithme *NSGA-II*, la surface de compromis pourra contenir au maximum le nombre d'individus de la simulation, dans notre cas 128. Si un nombre plus important de solutions efficaces est désiré, soit le nombre d'individus doit être augmenté, soit les solutions efficaces doivent être stockées dans une archive dont la taille évolue avec le nombre de solutions efficaces.

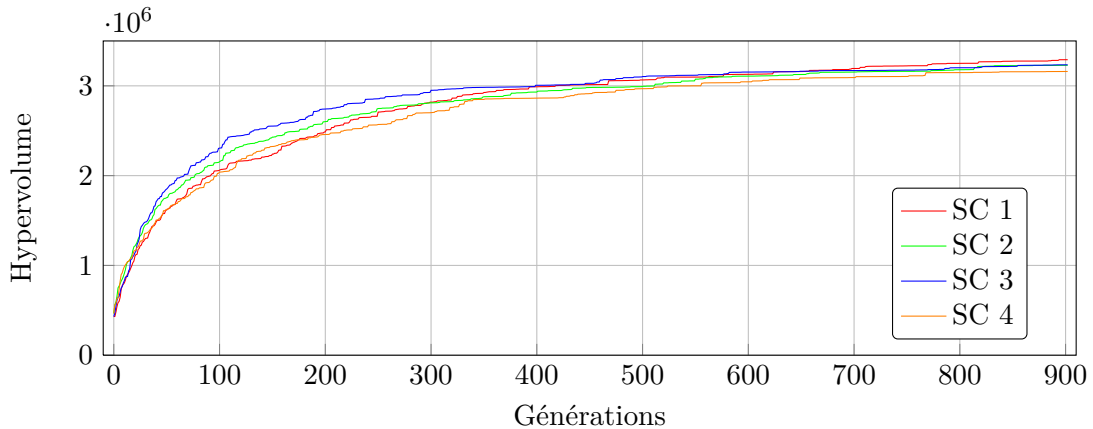


Figure 3.8 – Évolution de l'hypervolume pour quatre optimisations avec des populations initiales différentes pour le problème d'agencement de navires.

3.4.4 Analyse des résultats

Les résultats des quatre optimisations sont ici comparés. L'analyse de l'hypervolume a permis de détecter la convergence de l'algorithme d'optimisation. Présentée section 2.4.2.1, la métrique \mathcal{C} est maintenant utilisée pour comparer les surfaces de compromis entre elles. La table 3.3 présente les résultats de la métrique \mathcal{C} . Chaque case, représentée par ses indices i et j , contient la valeur de la métrique $\mathcal{C}(S_i, S_j)$, où S_i et S_j désignent respectivement les surfaces de compromis i et j . Comprise entre zéro et un, cette métrique exprime la proportion de points de S_j dominés par S_i . Le niveau de gris des cases permet d'analyser rapidement les résultats contenus dans cette table. Plus la case est

foncée, plus le résultat est proche de un. La diagonale de cette matrice correspond à la comparaison des différentes surfaces de compromis avec elles-mêmes. Étant donné qu'aucune solution ne se domine elle-même, la diagonale est remplie de zéro. En balayant ligne par ligne, on peut voir quelle surface de compromis se détache des autres. Les valeurs de la métrique \mathcal{C} pour la troisième surface de compromis sont toutes supérieures à 0.79, cela signifie que les points de cette surface domine la majorité des points des autres optimisations. Cette métrique exprime numériquement ce que l'on peut observer figure 3.6(c).

	1	2	3	4
1	0	0.9074	0.0625	0.5946
2	0.0000	0	0.0625	0.5405
3	0.7907	0.9074	0	0.9189
4	0.4186	0.5370	0.2813	0

Table 3.3 – Comparaison des surfaces de compromis à l'aide de la métrique \mathcal{C} .

Les figures 3.9 et 3.10 illustrent, à l'aide d'un calcul de densité de points, la manière dont les solutions de la première optimisation évoluent au cours des générations. Basée sur des calculs de distances entre points dans l'espace des objectifs, l'analyse de densité de points permet de visualiser les zones explorées par l'algorithme génétique. À chaque nouvelle génération calculée, un calcul de densité de répartition des points dans l'espace des solutions est réalisé. Plus la densité de points est importante, plus les couleurs sont chaudes. Enfin, les points cerclés de ces figures indiquent les points non dominés. La figure 3.9 montre la densité de répartition de la génération courante avec l'évolution de la surface de compromis. Les nouveaux points créés se partagent en deux catégories. La première catégorie correspond à l'ensemble des points améliorant la surface de compromis ou se trouvant proche d'elle. La deuxième catégorie correspond aux points éloignés de la surface de compromis avec un rang élevé. Les solutions associées à ces points ont peu de chance d'être retenues par la procédure élitiste de l'algorithme *NSGA-II* (Procédure présentée figure 2.10). La figure 3.10 présente l'évolution de la densité de répartition de l'ensemble des points calculés par l'algorithme génétique. Elle permet de visualiser les zones de l'espace des objectifs, où l'algorithme génétique a concentré ses efforts. Ces deux figures permettent d'observer visuellement la convergence du problème et de détecter une éventuelle stagnation de la convergence.

3.4.5 Extensions futures

Pour améliorer la modélisation actuelle, plusieurs choses peuvent être envisagées. On peut :

- intégrer des tolérances sur les aires des compartiments pour avoir un modèle plus souple et plus robuste.
- prendre en compte un ratio hauteur sur largeur minimum et maximum pour chaque compartiment. Ce ratio permettrait de s'assurer de la forme des compartiments, et d'éviter d'avoir des géométries trop allongées.
- établir des contraintes pour la localisation des compartiments. Actuellement, les compartiments sont libres de se placer sur tous les ponts. Le problème comporte trop de degrés de liberté ce qui explique que deux optimisations différentes proposent des solutions très différentes. De plus dans la pratique, le concepteur a quand même un certain nombre de contraintes de ce type. On peut envisager une répartition des compartiments pont par pont, c'est-à-dire attribuer un pont à chaque compartiment. Les variables de ce problème comporteraient autant de permutations que de ponts.
- prendre en compte davantage de critères pour représenter plus fidèlement les objectifs d'un agencement de navires. L'algorithme *NSGA-II* permet d'intégrer facilement d'autres objectifs. Toutefois au delà de trois objectifs, l'analyse des résultats deviendra plus compliquée : les surfaces de compromis devront être projetées pour être comparées.

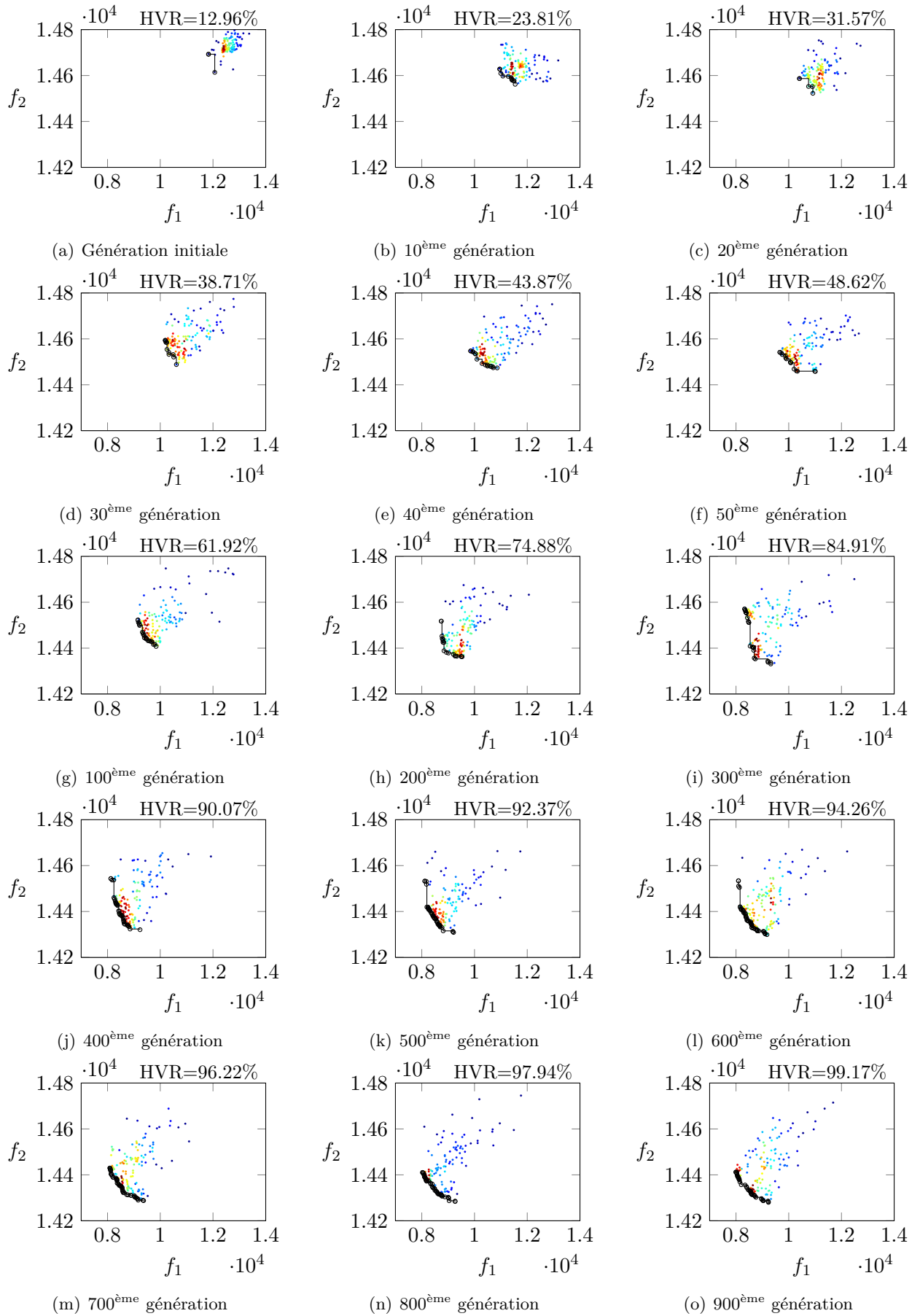


Figure 3.9 – Évolution de la surface de compromis pour le problème d’agencement de navire ainsi que de la répartition des points de la génération courante dans l’espace des objectifs. Plus les couleurs sont chaudes, plus la densité de points est élevée. Les points cerclés forment la surface de compromis de la génération courante.

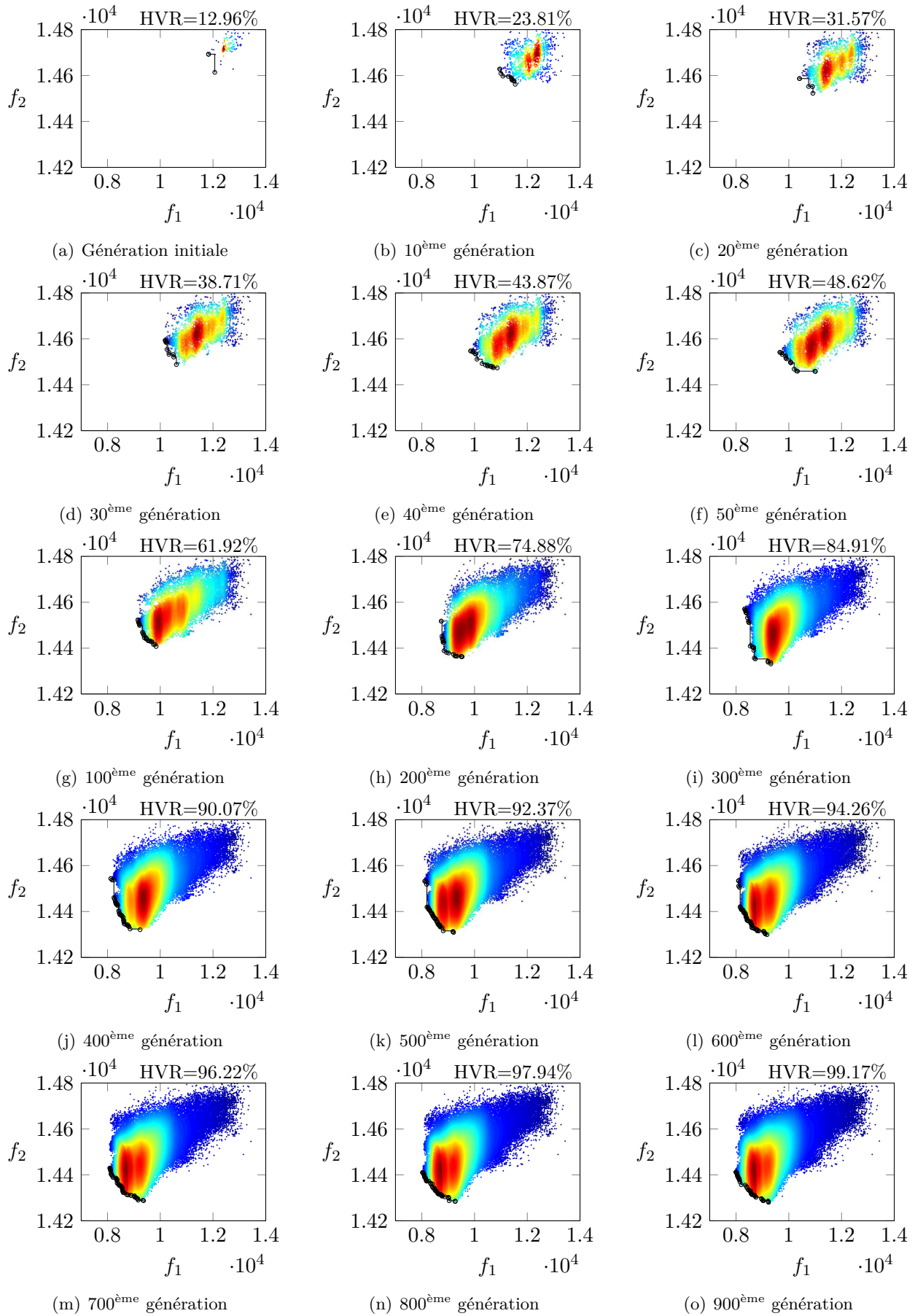


Figure 3.10 – Évolution de la surface de compromis pour le problème d'agencement de navire ainsi que de la répartition de l'ensemble des points calculés dans l'espace des objectifs. Plus les couleurs sont chaudes, plus la densité de points est élevée.

Enfin, les outils d'aide à la décision peuvent être utilisés pour choisir une solution parmi les solutions efficaces proposées par l'algorithme génétique. Yang *et al.* se sont intéressés à ces problèmes là pour les problèmes d'agencement multi-objectifs [YK03, YH07].

3.5 Conclusion

Ce chapitre a présenté la modélisation d'un problème d'agencement de navire ainsi que sa résolution multi-objectif. Pour s'assurer du respect des contraintes de placement, une méthode de placement légal a été utilisée. Cette méthode réalise un placement séquentiel des compartiments en remplissant méthodiquement les ponts. L'avantage de cette méthode réside dans sa simplicité de mise en œuvre, ainsi que dans son adaptation à la géométrie du navire. Le grand nombre de solutions possibles nous a conduit à utiliser un algorithme évolutionnaire stochastique. L'utilisation de l'algorithme multi-objectif *NSGA-II* nous a permis à l'aide d'une seule optimisation d'obtenir un ensemble de solutions efficaces. Le concepteur n'a pas été obligé d'exprimer de choix a priori, il peut exprimer ses choix en fonction des solutions proposées par l'optimiseur. L'utilisation des différents indicateurs multi-objectifs a permis de comprendre l'évolution de la convergence. L'analyse de l'hypervolume des différentes surfaces de compromis a révélé que le nombre initial de générations était trop faible. L'analyse des densités de point dans l'espace des objectifs fournit de bonnes indications quant à l'exploration de l'algorithme génétique.

Cet exemple d'agencement de navire est un cas particulier des problèmes d'agencement, où les compartiments étaient définis par une surface et non pas une géométrie. La modélisation de placement séquentiel peut être utilisée sur d'autres problèmes, comme les problèmes d'intégration à grande échelle ou encore les problèmes de chargement, ...

Même si cette méthode ne peut pas s'appliquer à l'ensemble des problèmes, l'exemple a permis d'explorer et de mettre au point bon nombre d'outils et de critères pour analyser les résultats d'agencement proposés. Ces outils seront, par la suite, utilisés dans le développement de la méthode de placement générique proposée au chapitre suivant.

Chapitre 4

Méthode de placement proposée

Ce chapitre présente la méthode de placement proposée pour la résolution de ces problèmes, ainsi que les différentes variantes de l'algorithme de séparation.

Sommaire

4.1	Introduction	60
4.2	Méthode de résolution des problèmes de placement proposée	60
4.2.1	Principe général	60
4.2.2	Modélisation du problème de placement bidimensionnel à orientations discrètes	63
4.2.3	Modélisation du problème de placement bidimensionnel à orientations continues	65
4.2.4	Modélisation du problème de placement tridimensionnel	65
4.2.5	Initialisation des solutions	66
4.3	Algorithmes de séparation	66
4.3.1	Algorithme de séparation en translation pour des composants parallélépipédiques	66
4.3.2	Algorithme de séparation en translation pour polygones	82
4.3.3	Algorithme de séparation pour des assemblages de cercles / sphères	87
4.3.4	Récapitulatif des différentes variantes de l'algorithme de séparation	100
4.4	Conclusion	100

4.1 Introduction

La littérature abonde de méthodes ad-hoc pour résoudre différents types de problèmes particuliers. Toutefois, la majorité des méthodes proposées ne permet pas d'intégrer de nouvelles contraintes ou encore est dédiée à des géométries particulières. Nous nous proposons de développer une méthode générique pour traiter le maximum de problèmes de placement. Le développement d'une telle méthode doit être capable de traiter les problèmes C&P ainsi que les problèmes d'agencement. De plus, elle doit être adaptée à tout type de géométrie, et être capable de prendre en compte tout type d'objectifs et de contraintes.

Nous avons choisi de développer une méthode de placement relaxé pour sa souplesse d'adaptation aux différents problèmes rencontrés. Les variables d'optimisation sont donc directement les variables de positionnement des composants. Les raisons qui ont motivé le choix d'une méthode de placement relaxé sont les suivantes :

- elle permet l'intégration de contraintes spécifiques dans la modélisation du problème, garantissant ainsi leur respect. L'alignement de composants, la mise à distance entre composants ou encore le positionnement spécifique des composants sont quelques-unes des contraintes qui peuvent être facilement gérées par une méthode relaxée.
- elle propose des solutions même si les contraintes ne sont pas entièrement satisfaites. Dans le cas où le problème est surcontraint, le concepteur peut aisément relaxer une partie des contraintes afin d'identifier une ou plusieurs solution(s) réalisable(s).
- enfin, elle permet au concepteur d'interagir facilement avec les solutions et de proposer des solutions qui lui semblent intéressantes.

4.2 Méthode de résolution des problèmes de placement proposée

La méthode proposée est présentée de façon modulaire. Tout d'abord, le principe général de la méthode est énoncé. Ensuite, les différentes versions de la méthode sont présentées. Les éléments numériques propres à chaque version font l'objet de la section suivante.

4.2.1 Principe général

Les problèmes de placement présentent une grande combinatoire avec de nombreux minima locaux. L'espace de recherche est très grand, voire infini lorsque le problème utilise des variables réelles. Pour explorer efficacement l'espace de recherche, un algorithme d'optimisation globale va être utilisé. Ce dernier sera stochastique pour permettre une exploration avec des temps de calcul raisonnable. Il pilotera les variables de positionnement de l'ensemble des composants avec leurs attributs et aura pour objectif de proposer des solutions efficaces. Les variables de position des composants sont choisies continues, les variables d'orientation des composants seront soit discrètes soit continues en fonction des degrés de liberté des composants.

La difficulté majeure associée à la méthode de placement relaxé réside dans le respect des contraintes de placement. Pour que les solutions proposées par l'algorithme d'optimisation globale soient réalisables, un algorithme de séparation est utilisé. Il a pour objectif de déplacer localement les composants afin que ceux-ci respectent au mieux les contraintes de placement.

L'utilisation d'algorithmes d'optimisation globale est quelque chose de classique dans la résolution des problèmes de placement. De nombreuses méthodes utilisant des algorithmes évolutionnaires ont été proposées [Gri98, TSqLL01, MFG08]. En revanche, le couplage d'un algorithme évolutionnaire et d'une méthode de séparation est quelque chose de nouveau.

Pour résumer, l'algorithme d'optimisation globale est chargé d'explorer efficacement l'espace de conception et l'algorithme de séparation a pour mission de rendre admissibles autant que possible les solutions proposées par l'optimiseur global. Les différents éléments nécessaires à la résolution des problèmes de placement sont présentés dans les paragraphes suivants.

4.2.1.1 Représentation géométrique et détection de collisions

En 2D et 3D, les composants peuvent être représentés de différentes manières. Le choix d'une représentation géométrique a pour objectif de faciliter les calculs géométriques notamment la détection de collisions. Cette dernière représente l'un des points durs de la résolution des problèmes de placement relaxé. La rapidité des algorithmes de détection de collision est cruciale pour la résolution des problèmes de placement. De leur efficacité dépendent les performances des méthodes de résolution proposées.

Les algorithmes de détection de collisions peuvent être classés selon trois niveaux : le premier consiste uniquement à détecter la collision, le second à quantifier la collision, enfin le troisième niveau consiste à utiliser les informations de collision pour séparer les composants se chevauchant. Plusieurs méthodes ont été mises en place pour quantifier les éventuelles collisions entre composants : la première, la plus évidente, consiste à calculer les aires / volumes d'intersection entre chaque paire de composants. Toutefois, cette méthode est trop coûteuse pour pouvoir être mise en place dans un cadre générique avec différentes géométries. Pour s'affranchir de cette difficulté, des alternatives ont été développées, comme par exemple avec l'utilisation des profondeurs de pénétration que nous présentons ci-après.

Notre objectif est de choisir une représentation géométrique adaptée à chaque problème, dont on pourra tirer partie pour la réalisation de l'algorithme de séparation. En 2D, ce choix va dépendre des degrés de liberté des composants, notamment de la libre rotation ou non des composants. En 3D, lorsque que les composants ont des géométries complexes, on s'oriente vers des assemblages de sphères, qui permettent une détection efficace des collisions.

4.2.1.2 Principe de l'algorithme de séparation

L'algorithme de séparation a pour mission de rendre admissible autant que possible les solutions proposées par l'optimiseur global. À partir d'une solution non admissible, l'algorithme cherche à minimiser une fonction caractérisant le non-respect des contraintes de placement en déplaçant localement l'ensemble des composants. L'objectif est de déplacer au minimum les composants de manière à perturber le moins possible la solution initiale proposée. La méthode est globale dans le sens où tous les composants sont déplacés en même temps pour minimiser le non-respect des contraintes de placement. Des heuristiques de séparation ont été proposées dans la littérature, toutefois elles déplacent les composants un par un et ne garantissent pas un déplacement minimum [ENB09]. Le problème de séparation de polygones est un problème \mathcal{NP} -complet même dans le cas où les composants sont des rectangles [LM95]. Il en sera donc de même en dimension supérieure pour des composants de géométrie complexe.

La fonction à minimiser s'écrit comme la somme de deux fonctions : la première traduit le non-respect des contraintes de non-chevauchement entre chaque paire de composants, la seconde exprime le non-respect des contraintes d'appartenance pour chaque composant. Si le problème de placement comporte m composants, alors la fonction à minimiser par l'algorithme de séparation s'écrit mathématiquement :

$$F(\mathbf{v}) = \underbrace{\omega_{\text{pen}} \sum_{i=1}^{m-1} \sum_{j=i+1}^m f_{ij}(\mathbf{v})}_{\text{Contraintes de non-chevauchement}} + \underbrace{\omega_{\text{pro}} \sum_{i=1}^m g_i(\mathbf{v})}_{\text{Contraintes d'appartenance}} \quad (4.1)$$

où \mathbf{v} est le vecteur position des m composants. Il contient généralement un vecteur de translation \mathbf{x} et un vecteur d'orientation \mathbf{o} . Les coefficients ω_{pen} et ω_{pro} sont des paramètres positifs permettant de régler de l'influence des deux fonctions. Les abréviations *pen* et *pro* correspondent respectivement aux termes anglophones *penetration* et *protrusion*, faisant référence aux contraintes de non-chevauchement et d'appartenance. Si aucune information n'est précisée, $\omega_{\text{pen}} = \omega_{\text{pro}} = 1$. Le terme $f_{ij}(\mathbf{v})$ traduit numériquement le non-respect de la contrainte entre les composants C_i et C_j . Si ces deux composants ne se chevauchent pas, alors cette valeur est nulle. Le terme g_i caractérise l'appartenance du composant C_i à l'intérieur du contenant. Si le composant C_i est situé intégralement dans le contenant, alors le terme $g_i(\mathbf{v})$ est nul. L'expression des fonctions f_{ij} et g_i dépend de la géométrie des composants, ainsi que de leurs degrés de libertés. Par la suite, ces fonctions seront choisies non linéaires de manière à

s'assurer de leur différentiabilité. Le problème de séparation est donc formulé comme un programme non linéaire non contraint.

La minimisation de la fonction F est assurée par la méthode quasi-Newton *BFGS* [Bro70, Fle70, Gol70, Sha70]. Les critères d'arrêt de cette méthode itérative sont au nombre de trois : un nombre maximum d'itérations, une précision sur la norme du gradient ainsi qu'un seuil au-dessous duquel l'amélioration de la valeur objectif n'est plus jugée suffisante. Les détails de cet algorithme sont présentés en annexe, section B.2. L'utilisation du gradient analytique de la fonction F permet d'éviter d'avoir recours à la différentiation numérique. Les détails de l'écriture de la fonction F et de son gradient sont présentés section 4.3. Les figures 4.9, 4.20, 4.29, présentées respectivement pages 76, 86 et 96, permettent de se représenter graphiquement le travail réalisé par les différentes variantes de l'algorithme de séparation. Les premières sous-figures correspondent aux solutions initiales, qui pourraient être fournies par l'optimiseur global. Les autres sous-figures montrent l'amélioration du respect des contraintes de placement.

4.2.1.3 Algorithme génétique multi-objectifs

L'optimiseur global choisi est l'algorithme génétique *Omni-Optimizer* [DT08]. Basé sur *NSGA-II* [DAPM00b], il est adapté à la résolution des problèmes mono-objectifs et multi-objectifs contraints. Sa prise en compte des distances phénotypiques et génotypiques, lui permet de conserver une diversité des solutions dans l'espace des objectifs et des variables. De plus, l'opérateur de sélection restreinte, qui effectue une sélection des meilleurs individus les plus proches avant le croisement des solutions, permet de trouver les solutions d'un problème multi-modal. L'algorithme génétique utilise classiquement les opérateurs de croisement continu (*Simulated Binary Crossover, SBX*) et de mutation polynomiale pour les variables réelles [DA95]. Le fonctionnement de ces opérateurs génétiques est présenté en annexe B.3.2.1 et B.3.2.2. Pour les variables discrètes, les opérateurs de croisement à deux points et de mutation par inversion de bit sont utilisés. Ce type de variables sera utilisé lorsque l'orientation des composants aura été choisie discrète.

Pour les problèmes présentant une forte compacité des composants, l'algorithme génétique requiert un grand nombre d'itérations avant de pouvoir converger. Pour pallier cet inconvénient, on introduit un nouvel opérateur qui échange la position de deux composants. Cette opération classique dans de nombreux optimiseurs de problèmes de placement, permet d'éviter la convergence trop rapide de l'algorithme génétique vers un minimum local. Tout comme les opérateurs de croisement et de mutation, cet opérateur agit avec une probabilité donnée.

4.2.1.4 Algorithme de placement proposé

L'algorithme proposé est présenté sur la figure 4.1. La structure de l'algorithme est très proche d'un algorithme génétique générationnel classique. La population initiale peut être générée de manière aléatoire, ou le concepteur peut fournir un ensemble de solutions de son choix ou encore des solutions obtenues avec des heuristiques de placement telles que « *Bottom-Left* » [Jak96]. Avant l'évaluation de chaque solution, les contraintes de placement sont vérifiées. Si ces dernières ne sont pas satisfaites, alors l'algorithme de séparation modifie la solution de manière à la rendre admissible. La solution sera admissible dès lors que la valeur de la fonction renvoyée par l'algorithme de séparation est nulle. Cette valeur sera par la suite prise en compte comme une contrainte de l'algorithme génétique : si la valeur de la fonction est nulle, alors les contraintes de placement sont satisfaites, dans le cas contraire, la solution modifiée n'est toujours pas admissible. Ces opérations sont répétées jusqu'à l'évaluation complète de la population. Si l'un des critères d'arrêt est satisfait, l'algorithme s'arrête. Sinon, les opérations de sélection par tournoi, de croisement, mutation et échange sont exécutées en vue de produire une nouvelle génération. Le critère d'arrêt peut être un nombre maximum de générations, une stagnation de l'évolution de l'hypervolume, ou tout autre critère défini par l'utilisateur.

Les sections suivantes présentent les modélisations des problèmes en fonction des géométries et degrés de liberté des composants.

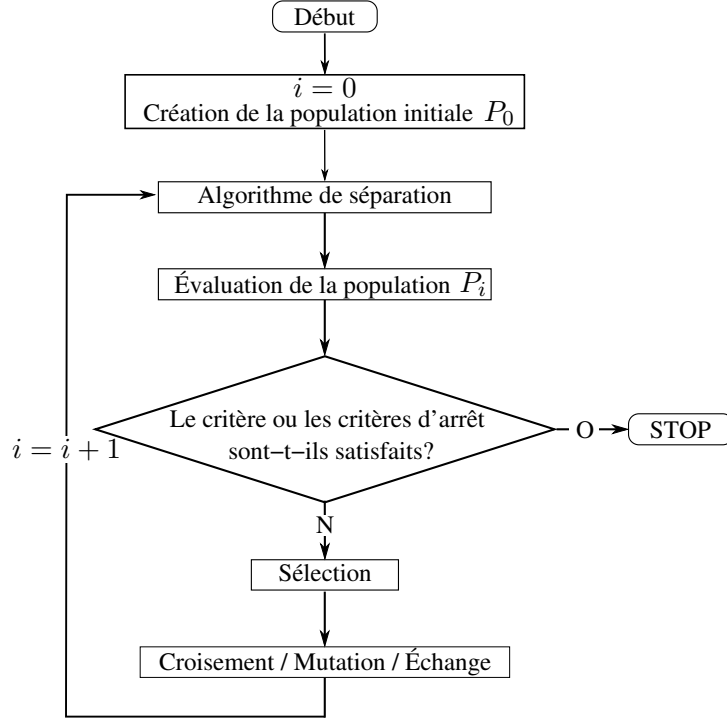


Figure 4.1 – Algorithme génétique proposé pour la résolution des problèmes de placement.

4.2.2 Modélisation du problème de placement bidimensionnel à orientations discrètes

Dans de nombreux problèmes 2D, les orientations des composants sont discrètes, c'est-à-dire choisies parmi un ensemble de valeurs définies par l'utilisateur. Dans la majorité des cas, ce sont des orientations orthogonales ($0^\circ, 90^\circ, 180^\circ, 270^\circ$). Ces orientations sont généralement dictées par la géométrie des composants. Toutefois, rien n'empêche, dans la version de l'algorithme présentée ici, de prendre en compte d'autres orientations. La modélisation des problèmes prenant en compte la libre rotation des composants fait l'objet de la section suivante.

Les composants sont représentés sous forme de polygones. Tous les composants, qui ne seraient pas sous cette forme là, sont approximés par des polygones. Les variables de position des composants sont continues, les variables d'orientation des composants sont discrètes. Les variables du composant C_i sont contenues dans le vecteur $\mathbf{v}_i = (\mathbf{x}_i, \theta_i) = (x_i, y_i, \theta_i)$, où θ_i est à choisir parmi l'ensemble d'orientations discrètes autorisées par le concepteur. Les composants sont pilotés à l'aide d'un point de référence choisi comme leur centre de gravité. La position du polygone P_i est calculée à l'aide d'une composition d'une rotation et d'une translation. Cette position se calcule de la manière suivante :

$$\begin{pmatrix} P_{ij,x}^n \\ P_{ij,y}^n \end{pmatrix} = \mathbf{c}_{ij}(\mathbf{v}_i) = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \begin{pmatrix} P_{ij,x} \\ P_{ij,y} \end{pmatrix} \quad (4.2)$$

où $P_{ij,x}$ et $P_{ij,y}$ représentent les coordonnées initiales en abscisses et ordonnées du point j du polygone P_i . Les nouvelles coordonnées de ce point sont notées $P_{ij,x}^n$ et $P_{ij,y}^n$. La figure 4.2 illustre les calculs réalisés pour positionner chaque composant i . On applique tout d'abord la rotation θ_i autour de son centre de gravité, puis le composant est translaté du vecteur \mathbf{x}_i .

Comme mentionné précédemment, le calcul des aires d'intersection des composants pour la détection de collision est trop coûteux. D'autres idées ont donc été développées, comme celles basées sur l'utilisation des polygones de non-recouvrement (*No-Fit Polygon*) et des polygones d'appartenance (*Inner-Fit Polygon*). Ces polygones sont présentés figure 4.3 pour deux polygones P_1 et P_2 . Le polygone P_1 est fixe et la position du polygone P_2 est repérée à l'aide d'un point de référence, matérialisé

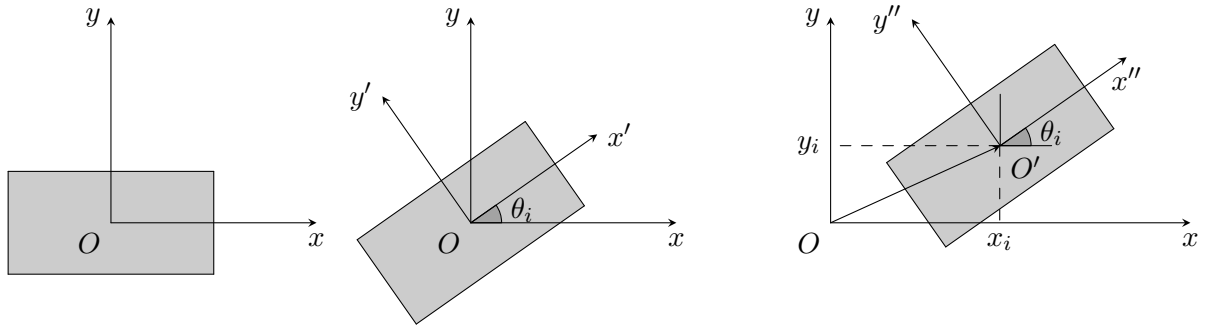
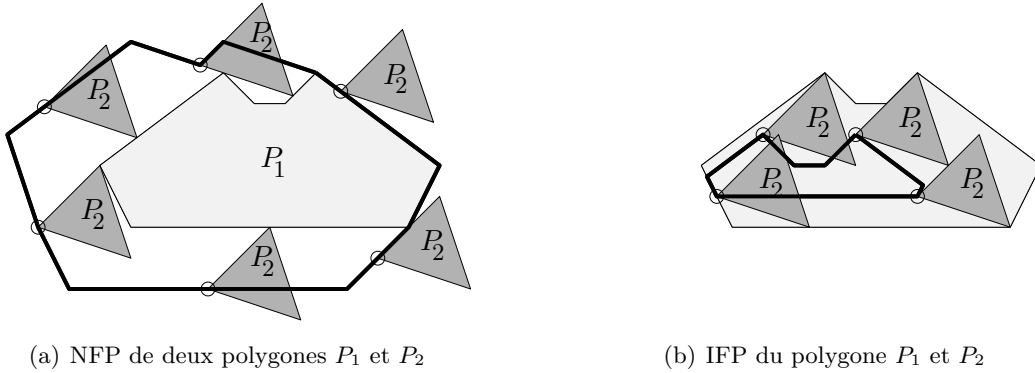


Figure 4.2 – Composition de transformations pour le placement 2D des composants.



(a) NFP de deux polygones P_1 et P_2

(b) IFP du polygone P_1 et P_2

Figure 4.3 – Calcul des polygones de non-recouvrement et d'appartenance. Ces deux polygones sont représentés en trait épais noir sur les deux sous-figures.

sur la figure par le cercle noir. Le polygone de non-recouvrement $NFP(P_1, P_2)$ permet de savoir si P_1 et P_2 se chevauchent en analysant la position du point de référence de P_2 . Si le point de référence de P_2 est à l'intérieur de $NFP(P_1, P_2)$, alors P_1 et P_2 se chevauchent. Si le point de référence de P_2 est situé sur le polygone de non-recouvrement $NFP(P_1, P_2)$, alors P_1 et P_2 sont en contact. Dans le cas contraire, P_1 et P_2 ne sont pas en collision (Figure 4.3(a)). Le même raisonnement peut être appliqué pour tester l'appartenance d'un polygone à un autre polygone : le polygone P_2 est contenu dans le polygone P_1 , si le point de référence de P_2 est situé à l'intérieur du polygone d'appartenance (Figure 4.3(b)).

Même si le calcul des polygones de non-recouvrement et d'appartenance est coûteux et doit être effectué pour chaque orientation des polygones, l'utilisation de cette méthode présente deux avantages :

- les collisions peuvent être détectées par un simple test d'appartenance d'un point à un polygone. Ce test présente une complexité linéaire avec le nombre de segments composant le polygone ;
- d'autre part, le test d'appartenance permet de récupérer des informations (profondeurs de pénétration) qui seront utiles à l'algorithme de séparation pour évaluer numériquement le chevauchement de deux composants.

Le calcul de ces polygones est basé sur l'utilisation des sommes de Minkowski [HYB07]. Dans le cas où les polygones sont convexes, leur somme de Minkowski est un polygone convexe, dont la complexité de calcul est linéaire avec le nombre de sommets. Dans le cas où les polygones sont quelconques, une décomposition préliminaire en un assemblage de polygones convexes est nécessaire. L'ensemble des polygones de non-recouvrement et des polygones d'appartenance est calculé lors d'un prétraitement initial. Les différentes techniques de calcul des polygones de non-recouvrement et d'appartenance sont présentées en annexe section B.4.

Avec une orientation discrète des composants, l'algorithme de séparation se base sur ces polygones de non-recouvrement et appartenance pour détecter la collision entre polygones et établir les directions de séparation entre composants. Présentée section 4.3.2, la séparation consistera à traduire les composants pour obtenir une solution réalisable.

4.2.3 Modélisation du problème de placement bidimensionnel à orientations continues

Lorsque l'orientation des composants n'est plus discrète mais continue, la détection de collisions basée sur l'utilisation des polygones de non-recouvrement et appartenance n'est plus performante. En effet, il faut calculer ces polygones pour chaque nouvelle orientation. Même si ce calcul était immédiat, la version de l'algorithme de séparation précédemment présentée translate uniquement les composants. L'optimisation des variables d'orientation serait donc laissée à l'optimiseur global. Ce dernier a peu de chance de proposer par lui-même la bonne combinaison d'orientations pour satisfaire les contraintes de placement.

Pour toutes ces raisons, on choisit de changer de représentation géométrique pour les composants à positionner. Nous nous orientons vers une représentation basée sur des assemblages de cercles, pour lesquels un algorithme de séparation en translation et rotation peut être élaboré. Pour des raisons de convergence de l'algorithme de séparation, les assemblages de cercles sont construits à partir des axes médians des composants [Att95, TV04]. L'axe médian ou squelette d'un composant correspond à l'ensemble des centres de ses boules maximales. Il peut être vu comme la version continue du diagramme de Voronoï. La figure 4.4 présente un calcul d'axe médian pour une forme quelconque, dans laquelle quelques cercles de rayon maximal ont été tracés. L'algorithme de séparation associé est présenté section 4.3.3.

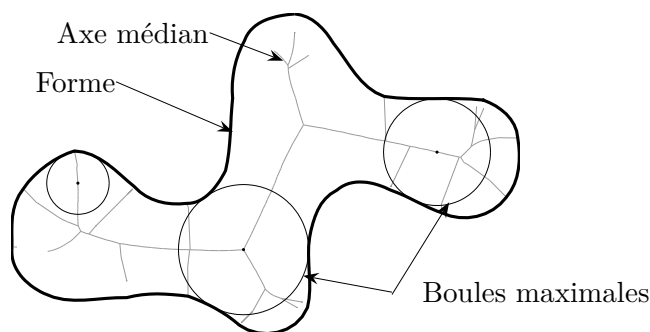


Figure 4.4 – Axe médian et boules maximales

4.2.4 Modélisation du problème de placement tridimensionnel

Avec des composants de géométrie complexe, la modélisation des problèmes tridimensionnels est une généralisation de la modélisation du placement bidimensionnel à orientations continues. L'utilisation des géométries exactes ne permet pas une détection rapide de collisions et n'est pas adaptée aux algorithmes de séparation. On choisit donc de représenter les composants 3D à l'aide d'assemblages de sphères, qui nous permettront de détecter facilement les intersections entre composants. De plus, les éventuelles collisions pourront être exploitées pour réaliser la séparation des composants se chevauchant. Les techniques de décomposition en sphères sont détaillées lors de la présentation de l'algorithme de séparation associé.

Dans le cas général, les variables de positionnement de chaque composant sont au nombre de six. Trois sont utilisées pour traduire le composant et trois autres servent à calculer son orientation. On choisit les angles d'Euler $z - x - z$ pour représenter cette orientation. Le calcul de la matrice de rotation pour les angles d'Euler est présenté en annexe B.5.

Si tous les composants et contenant sont des parallélépipèdes rectangles à orientations orthogonales, il n'est pas nécessaire de transformer les composants. Dans cette situation, une modélisation et une version spécifique de l'algorithme de séparation sont proposées section 4.3.1. Dans le cas général, les algorithmes de séparation 3D sont présentés sections 4.3.3.2 et 4.3.3.3.

4.2.5 Initialisation des solutions

L'initialisation de la population de l'algorithme génétique va jouer un rôle important sur la vitesse de convergence de l'algorithme, ainsi que sur la diversité génotypique des solutions finales. De plus, les résultats présentés dans le chapitre 5 montrent qu'il peut être difficile de trouver un ensemble de solutions réalisables. Pour pallier ces inconvénients, on se propose d'initialiser la population de l'algorithme global avec des solutions réalisables.

La première idée qui vient à l'esprit, consiste à générer des solutions réalisables à partir de différentes optimisations et à utiliser ces solutions réalisables comme population initiale. Il faut pour cela exécuter plusieurs fois la méthode proposée, sauver les premières solutions réalisables identifiées, et relancer la méthode avec une population initiale différente autant de fois que nécessaire pour obtenir une population de solutions réalisables. Cette technique peut être coûteuse pour les problèmes où une solution réalisable est difficile à identifier.

La seconde idée, que nous avons retenue, consiste à utiliser une méthode dédiée. En 2D, nous avons donc développé une méthode de placement légal, qui génère des solutions satisfaisant les contraintes de non-chevauchement et d'appartenance. La méthode est une heuristique de type *Bottom-Left-Fill* adaptée aux composants et contenants de géométrie complexe. Cette heuristique assure un placement séquentiel des composants qui garantit le respect des contraintes de placement. Une permutation aléatoire est utilisée pour représenter l'ordre d'introduction des composants, les orientations des composants sont elles aussi choisies aléatoirement parmi les différentes orientations possibles. Pour chaque composant à placer, son polygone d'appartenance est calculé. Son point le plus bas et le plus à gauche est cherché. Sur ce point va être placé le composant, qui va par la suite faire partie du contenant. Le procédé est répété jusqu'à ce que l'ensemble des composants soit positionné, ou bien qu'il n'y ait plus de place dans le contenant. Pour varier les solutions proposées, les directions de dépôt de l'heuristique peuvent être interverties : par exemple, les composants peuvent être placés en priorité en haut à droite. Enfin, un algorithme d'optimisation stochastique travaillant sur la permutation gérant l'ordre d'introduction des composants peut être utilisé pour les situations où l'initialisation est difficile. Avec cette seconde méthode, tous les composants seront en contact. Elle sera donc bien adaptée aux problèmes à forte compacité, pour lesquels les composants des solutions optimales seront en contact. Cependant, les contraintes de placement spécifiques, comme l'alignement de composants, ne sont pas actuellement gérées. La figure 4.5 illustre le fonctionnement de cette heuristique sur une instance de problèmes de découpe de formes irrégulières. Les composants sont ici placés d'abord le plus à gauche, puis le plus en bas : l'heuristique utilisée ici porte donc le nom *Left-Bottom-Fill*.

Pour les problèmes tridimensionnels, l'adaptation de l'heuristique *Bottom-Left-Fill* basée sur une représentation en *voxels* de Tiwari *et al.* [TFF08] pourra être utilisée pour initialiser les solutions.

Maintenant que les différentes techniques de modélisation des problèmes ont été présentées, les différentes versions de l'algorithme de séparation sont présentées en détails.

4.3 Algorithmes de séparation

L'algorithme de séparation est utilisé pour faire respecter les contraintes de placement du problème (contraintes de non-chevauchement entre les composants et contraintes d'appartenance des composants au contenant). L'idée générale consiste à regrouper l'ensemble des contraintes de placement dans une seule et même fonction continue et différentiable. Ensuite, l'algorithme d'optimisation déterministe *BFGS* est utilisé pour minimiser ce non-respect des contraintes. Cet algorithme peut être décliné sous plusieurs versions en fonction des représentations géométriques choisies.

4.3.1 Algorithme de séparation en translation pour des composants parallélépipédiques

Dans le cas où l'ensemble des composants et contenant sont des parallélépipèdes rectangles alignés sur le système d'axes, une version spécifique de l'algorithme de séparation peut être développée.

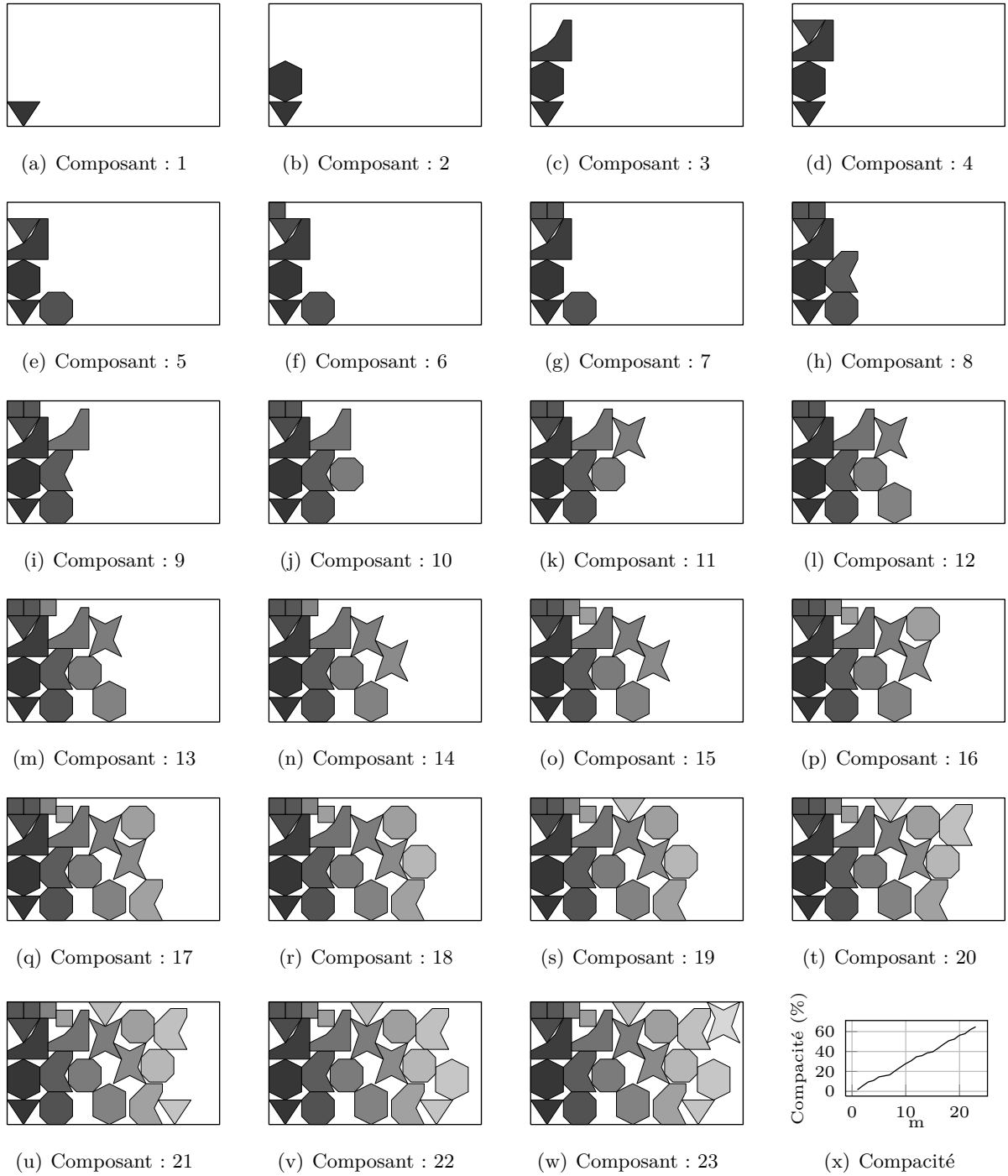


Figure 4.5 – Déroulement l'heuristique de placement *Left-Bottom-Fill* sur l'instance *Blaz* [BHW93]. Chaque composant est placé un à un en fonction de la permutation indiquant l'ordre d'introduction des composants. L'heuristique fait appel aux fonctions géométriques de calcul de polygones non-recouvrement et d'appartenance. Le niveau de gris des composants indique l'ordre d'arrivée des composants : plus ils sont foncés, plus ils sont arrivés tôt.

De nombreuses méthodes de placement ont été développées pour les rectangles et parallélépipèdes, toutefois elles ne permettent pas de prendre en compte des contraintes de placement spécifiques, comme l'alignement de certains composants. L'algorithme présenté ici est développé pour prendre en compte ces contraintes.

Nous nous proposons de chercher une fonction de minimisation F caractérisant le non-respect de contraintes de placement, qui sera la plus efficace pour séparer un ensemble de parallélépipèdes. Pour cela, différentes fonctions test sont élaborées et comparées en vue de choisir la meilleure. Deux grandeurs sont à la base de toute caractérisation de ce non-respect : la largeur des intervalles d'intersection de segments et la profondeur de pénétration. Ces deux notions sont présentées pour le cas 1-D dans le premier paragraphe puis généralisées en dimension supérieure dans le second. Ces fonctions élémentaires sont ensuite utilisées pour construire les différentes fonctions test. Plusieurs tests sont réalisés dans le quatrième paragraphe pour déterminer la fonction de minimisation la plus efficace. La prise en compte des rotations est expliquée dans le cinquième paragraphe. Enfin, les différentes extensions pour l'intégration de nouvelles contraintes dans l'algorithme sont présentées.

4.3.1.1 Cas unidimensionnel

Dans un premier temps, on considère un ensemble de m segments libres de se translater suivant un axe. On souhaite évaluer la contrainte de non-chevauchement entre chaque paire de segments.

Le point de référence de ces segments, qui sera utilisé comme variable d'optimisation, est choisi comme le centre de ces segments. Ce choix, qui apparaît de prime abord plus coûteux, s'avère plus judicieux lors de la prise en compte des rotations discrètes en dimension supérieure. Les justifications seront données dans la section suivante.

Soient \mathbf{x} le vecteur position de l'ensemble des m segments, et \mathbf{b} le vecteur donnant la longueur de chacun de ces segments ($\mathbf{x} \in \mathbb{R}^m, \mathbf{b} \in \mathbb{R}^m$). On commence par évaluer la contrainte de non-chevauchement pour deux segments i et j . Les positions de ces deux segments sont données par les variables x_i et x_j . Deux segments i et j s'intersectent si et seulement si il existe un point appartenant à ces deux segments. L'ensemble de ces points forme un segment d'intersection. La fonction γ_{ij} caractérise la largeur de l'intervalle d'intersection des segments i et j .

$$\gamma_{ij}(\mathbf{x}) = \max \{0, \min \{x_i + b_i/2, x_j + b_j/2\} - \max \{x_i - b_i/2, x_j - b_j/2\}\} \quad (4.3)$$

La profondeur de pénétration δ_{ij} entre deux segments i et j est définie comme la distance minimale de translation d'un des segments pour que ces segments ne se chevauchent plus. Cette profondeur δ_{ij} s'écrit mathématiquement :

$$\delta_{ij}(\mathbf{x}) = \max \{0, b_i/2 + b_j/2 - |x_i - x_j|\} \quad (4.4)$$

Ces deux fonctions sont linéaires définies par morceaux. Si les deux segments i et j ne partagent aucun point en commun, alors ces deux fonctions renvoient la valeur nulle. La différence majeure entre ces deux fonctions se situe au niveau de leur gradient, comme il le sera expliqué par la suite. La figure 4.6 présente deux segments se chevauchant et pour lesquels les profils des fonctions γ_{ij} et δ_{ij} sont tracés. Le segment j est considéré fixe et le segment i est libre de se translater suivant l'axe x .

L'équation 4.5 donne l'expression de la fonction γ_{ij} si on avait choisi comme point de référence l'extrémité gauche de chaque segment. L'équation 4.6 donne respectivement l'expression de γ_{ij} pour l'extrémité droite.

$$\gamma_{ij}(\mathbf{x}) = \max \{0, \min \{x_i + b_i, x_j + b_j\} - \max \{x_i, x_j\}\} \quad (4.5)$$

$$\gamma_{ij}(\mathbf{x}) = \max \{0, \min \{x_i, x_j\} - \max \{x_i - b_i, x_j - b_j\}\} \quad (4.6)$$

Si on travaille avec les demi-longueurs des segments, le surcoût de l'équation 4.3 n'est que deux additions par rapport aux équations 4.5 et 4.6. Le choix du point de référence pour le calcul de la profondeur de pénétration δ_{ij} n'influence pas la complexité de calcul de cette fonction. Dans l'implémentation des algorithmes, on travaillera avec les demi-longueurs des segments pour éviter à chaque fois les divisions par deux de l'équation 4.3.

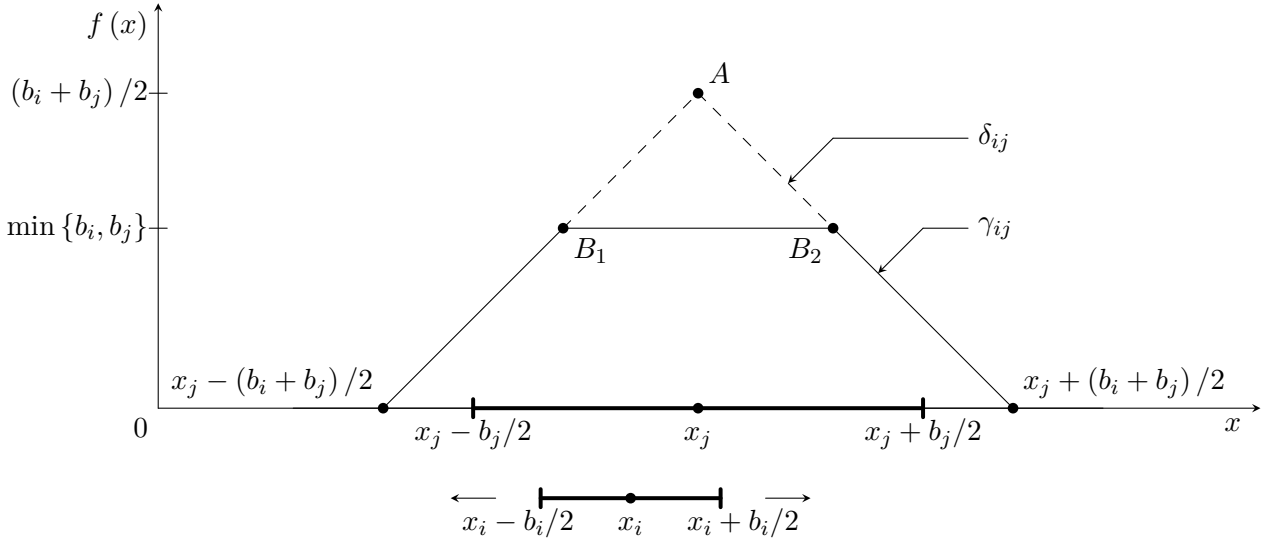


Figure 4.6 – Profils des fonctions γ_{ij} et δ_{ij} . Le segment j est considéré fixe et le segment i peut être translaté. L'axe x représente donc la coordonnée du centre du segment i . Les points A , B_1 , et B_2 ont pour coordonnées respectives $(x_j, (b_i + b_j)/2)$, $(x_j - b_j/2 + b_i, \min\{b_i, b_j\})$ et $(x_j + b_j/2 - b_i, \min\{b_i, b_j\})$.

Les gradients des fonctions γ_{ij} et δ_{ij} , qui seront utilisés pour réaliser l'algorithme de séparation, s'expriment de la manière suivante :

$$\begin{aligned} \frac{\partial \gamma_{ij}(\mathbf{x})}{\partial x_i} &= \begin{cases} 1 & \text{si } x_j - b_j/2 - b_i/2 < x_i < x_j - b_j/2 + b_i/2, \\ -1 & \text{si } x_j + b_j/2 - b_i/2 < x_i < x_j + b_j/2 + b_i/2, \\ 0 & \text{sinon.} \end{cases} \\ \frac{\partial \delta_{ij}(\mathbf{x})}{\partial x_i} &= \begin{cases} 1 & \text{si } x_j - (b_i + b_j)/2 < x_i < x_j, \\ -1 & \text{si } x_j < x_i < x_j + (b_i + b_j)/2, \\ 0 & \text{sinon.} \end{cases} \end{aligned} \quad (4.7)$$

La fonction γ_{ij} présente un gradient nul lorsque qu'un des deux segments est entièrement contenu dans l'autre. Ce phénomène peut causer une convergence prématurée des algorithmes d'optimisation. En revanche, la fonction δ_{ij} présente un gradient non nul dans ce cas de figure.

Dans le cas où les composants et le contenant sont des segments, rectangles ou parallélépipèdes, les contraintes d'appartenance peuvent être prises en compte de deux manières : La première repose sur une fonction de pénalité. La seconde consiste à utiliser un optimiseur sachant gérer les bornes des variables, comme *L-BFGS-B* [BLNZ95, BNZ97]. Les bornes permettent de contraindre les composants à rester dans le contenant. Toutefois, la généralisation des contraintes d'appartenance pour des contenants de géométrie quelconque ne pourra pas être modélisée à l'aide de bornes. De plus, les rotations orthogonales de rectangles ou de parallélépipèdes nécessitent de mettre à jour les bornes des variables d'optimisation pour chaque rotation. Nous avons donc choisi d'utiliser une fonction de pénalité pour gérer les contraintes d'appartenance. Un seul et même algorithme d'optimisation (*BFGS*) sera donc utilisé pour réaliser les séparations.

Pour modéliser la contrainte d'appartenance du segment i au contenant, on introduit deux nouvelles fonctions $\bar{\gamma}_i$ et $\bar{\delta}_i$ qui s'apparentent aux complémentaires des fonctions γ_{ij} et δ_{ij} . Le contenant est un segment de longueur c , dont l'extrémité gauche est placée à l'origine du repère. L'ensemble des segments est supposé avoir une longueur inférieure à celle du contenant. La fonction $\bar{\gamma}_i$ évalue la longueur du segment situé à l'extérieur du contenant. La fonction $\bar{\delta}_i$ quantifie la translation nécessaire pour que le segment i soit intégralement dans le contenant.

$$\begin{aligned} \bar{\gamma}_i(\mathbf{x}) &= b_i - \max\{0, \min\{c, x_i + b_i/2\} - \max\{0, x_i - b_i/2\}\} \\ \bar{\delta}_i(\mathbf{x}) &= \max\{0, |x_i - c/2| + (b_i - c)/2\} \end{aligned} \quad (4.8)$$

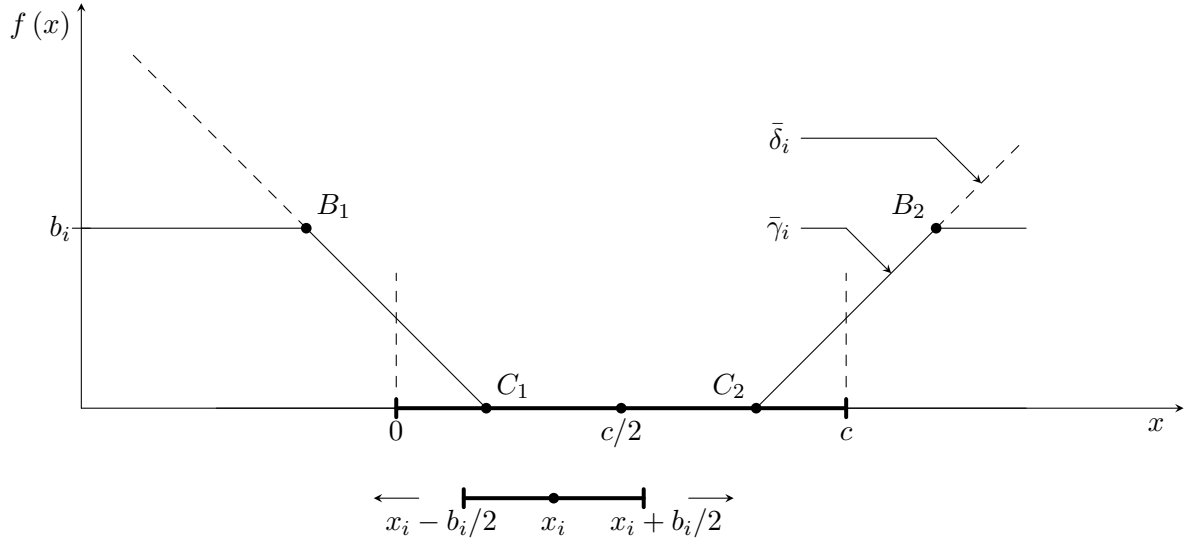


Figure 4.7 – Profil des fonctions $\bar{\delta}_i$ et $\bar{\gamma}_i$. Le contenant de longueur c est considéré fixe et le segment i peut être traduit. L'axe x représente donc la coordonnée du centre du segment i . Les points B_1 et B_2 ont pour coordonnées respectives $(-b_i/2, b_i)$ et $(c + b_i/2, b_i)$. Les coordonnées des points C_1 et C_2 sont respectivement $(b_i/2, 0)$ et $(c - b_i/2, 0)$.

La figure 4.7 présente les fonctions $\bar{\gamma}_i$ et $\bar{\delta}_i$. Les dérivées de ces fonctions sont présentées équation 4.9.

$$\begin{aligned} \frac{\partial \bar{\gamma}_i(\mathbf{x})}{\partial x_i} &= \begin{cases} 1 & \text{si } c - b_i/2 < x_i < c + b_i/2, \\ -1 & \text{si } -b_i/2 < x_i < b_i/2, \\ 0 & \text{sinon.} \end{cases} \\ \frac{\partial \bar{\delta}_i(\mathbf{x})}{\partial x_i} &= \begin{cases} 1 & \text{si } c - b_i/2 < x_i, \\ -1 & \text{si } x_i < b_i/2, \\ 0 & \text{sinon.} \end{cases} \end{aligned} \quad (4.9)$$

Les fonctions γ_{ij} , δ_{ij} , $\bar{\gamma}_i$ et $\bar{\delta}_i$ peuvent être utilisées pour générer des fonctions traduisant le non-respect des contraintes de placement dans un problème 1D. Elles peuvent être aussi utilisées pour une généralisation du problème à plusieurs dimensions.

4.3.1.2 Cas multidimensionnel

On présente ici une généralisation du cas unidimensionnel. Soit d la dimension du problème. Si $d = 2$, les composants considérés seront des rectangles alignés sur le système d'axe. Si $d = 3$, les composants considérés seront des parallélépipèdes rectangles. Dans le cas général, on parlera de boîtes alignées sur le système d'axe. Il est implicitement sous-entendu que l'ensemble des boîtes est aligné sur le système d'axe.

Le pilotage des boîtes se fait à l'aide de leur centre. Les dimensions des boîtes sont définies à l'aide d'un vecteur de taille d , où chaque composante se reporte à la dimension associée. Par la suite, l'indice k sera utilisé pour se référer à la dimension courante. Une boîte i est entièrement définie par deux vecteurs : un vecteur de dimension et un vecteur de position. Soit \mathbf{x} la matrice de position des centres des boîtes. Cette matrice possède m lignes, et d colonnes ($\mathbf{x} \in \mathbb{R}^{m \times d}$). La ligne i représente les coordonnées du centre de la boîte i . La colonne k représente les coordonnées des centres des boîtes suivant la dimension k . La variable x_{ik} représente donc la coordonnées de la $i^{\text{ème}}$ boîte suivant la dimension k . Soit \mathbf{b} la matrice de dimension des boîtes ($\mathbf{b} \in \mathbb{R}^{m \times d}$). Cette matrice a la même taille que la matrice \mathbf{x} , et le système d'indices est identique. Soit \mathbf{c} le vecteur de dimension du contenant. Ce vecteur possède autant de composantes que le problème a de dimensions ($\mathbf{c} \in \mathbb{R}^d$). Comme précédemment, l'hypothèse est faite que l'ensemble des boîtes puissent être contenues dans le contenant, c'est-à-dire $0 < b_{ik} < c_k, \forall i \in \{1, \dots, m\}, \forall k \in \{1, \dots, d\}$.

Les développements présentés ici sont basés sur le résultat suivant : Soient deux boîtes i et j de dimension d alignées sur le système d'axe. Ces deux boîtes sont parfaitement définies à l'aide des deux vecteurs position \mathbf{x}_i , \mathbf{x}_j ainsi que des vecteurs dimension \mathbf{b}_i , \mathbf{b}_j . Pour chaque dimension k , il est possible de calculer les largeurs des segments d'intersection ainsi que les profondeurs de pénétration. Dès lors qu'une de ces grandeurs est nulle, les boîtes i et j ne se chevauchent pas. En revanche, si l'ensemble de ces grandeurs est différent de zéro, les boîtes i et j se chevauchent.

On commence par généraliser les quatre fonctions γ_{ij} , δ_{ij} , $\bar{\gamma}_i$ et $\bar{\delta}_i$ aux différentes dimensions du problème. Les fonctions γ_{ijk} , δ_{ijk} , $\bar{\gamma}_{ik}$ et $\bar{\delta}_{ik}$, qui se rapportent à la $k^{\text{ième}}$ dimension du problème, s'écrivent :

$$\begin{aligned}\gamma_{ijk}(\mathbf{x}) &= \max \{0, \min \{x_{ik} + b_{ik}/2, x_{jk} + b_{jk}/2\} - \max \{x_{ik} - b_{ik}/2, x_{jk} - b_{jk}/2\}\} \\ \delta_{ijk}(\mathbf{x}) &= \max \{0, (b_{ik} + b_{jk})/2 - |x_{ik} - x_{jk}|\} \\ \bar{\gamma}_{ik}(\mathbf{x}) &= b_{ik} - \max \{0, \min \{c_k, x_i + b_{ik}/2\} - \max \{0, x_{ik} - b_{ik}/2\}\} \\ \bar{\delta}_{ik}(\mathbf{x}) &= \max \{0, |x_{ik} - c_k/2| + (b_{ik} - c_k)/2\}\end{aligned}\tag{4.10}$$

Les gradients de ces quatre fonctions sont similaires aux gradients présentés aux équations 4.7 et 4.9, seul l'indice k sur la dimension a été ajouté.

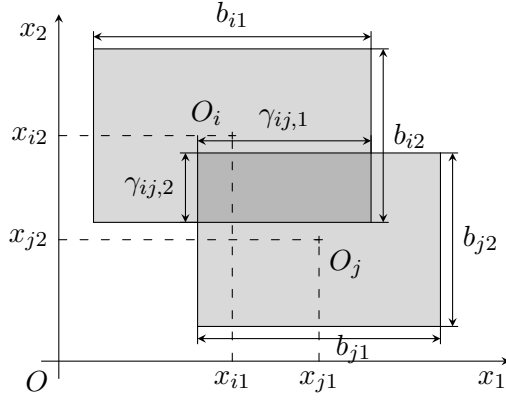
On introduit quatre nouvelles fonctions Γ_{ij} , Δ_{ij} , $\bar{\Gamma}_i$ et $\bar{\Delta}_i$ qui traduisent respectivement le comportement global des fonctions γ_{ijk} , δ_{ijk} , $\bar{\gamma}_{ik}$ et $\bar{\delta}_{ik}$. La fonction Γ_{ij} s'écrira comme le produit des fonctions γ_{ijk} sur l'ensemble des dimensions du problème, et Δ_{ij} caractérisera la distance minimale de translation des composants pour que ceux-ci ne se chevauchent plus. La fonction $\bar{\Gamma}_i$ évalue la portion de la boîte i ne se trouvant pas dans le contenant. Enfin, la fonction $\bar{\Delta}_i$ évalue la translation nécessaire pour que la boîte i soit dans le contenant.

$$\begin{aligned}\Gamma_{ij}(\mathbf{x}) &= \prod_{k=1}^d \gamma_{ijk}(\mathbf{x}) \\ &= \prod_{k=1}^d \max \{0, \min \{x_{ik} + b_{ik}/2, x_{jk} + b_{jk}/2\} - \max \{x_{ik} - b_{ik}/2, x_{jk} - b_{jk}/2\}\} \\ \Delta_{ij}(\mathbf{x}) &= \min_{k \in \{1, \dots, d\}} \{\delta_{ijk}(\mathbf{x})\} \\ &= \min_{k \in \{1, \dots, d\}} \{\max \{0, (b_{ik} + b_{jk})/2 - |x_{ik} - x_{jk}|\}\} \\ \bar{\Gamma}_i(\mathbf{x}) &= \prod_{k=1}^d b_{ik} - \prod_{k=1}^d \max \{0, \min \{c_k, x_i + b_{ik}/2\} - \max \{0, x_{ik} - b_{ik}/2\}\} \neq \prod_{k=1}^d \bar{\gamma}_{ik}(\mathbf{x}) \\ \bar{\Delta}_i(\mathbf{x}) &= \max_{k \in \{1, \dots, d\}} \{\bar{\delta}_{ik}(\mathbf{x})\} \\ &= \max_{k \in \{1, \dots, d\}} \{\max \{0, |x_{ik} - c_k/2| + (b_{ik} - c_k)/2\}\}\end{aligned}\tag{4.11}$$

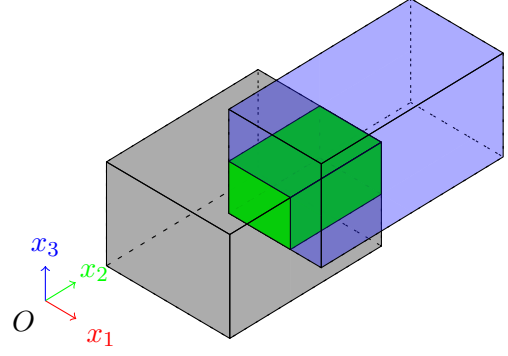
En 2D, Γ_{ij} traduit l'aire d'intersection des rectangles i et j , $\bar{\Gamma}_i$ évalue l'aire de la partie du rectangle i située à l'extérieur du contenant. En 3D, Γ_{ij} exprime le volume d'intersection des parallélépipèdes i et j et $\bar{\Gamma}_i$ le volume situé à l'extérieur du contenant. La figure 4.8 présente les intersections de deux rectangles et deux parallélépipèdes. La sous-figure 4.8(a) reprend la notation utilisée ci-avant.

Les calculs des gradients de ces fonctions, qui seront utilisés lors des différentes optimisations, sont maintenant présentés. La notation ∇_p sera utilisée pour désigner le vecteur gradient se rapportant au composant p : $\nabla_p = (\partial/\partial x_{p1}, \dots, \partial/\partial x_{pd})$. Tout d'abord, l'ensemble des gradients nuls est listé :

$$\begin{aligned}\nabla_p \Gamma_{ij}(\mathbf{x}) &= \mathbf{0} \quad p \in \{1, \dots, m\} \setminus \{i, j\} \\ \nabla_p \Delta_{ij}(\mathbf{x}) &= \mathbf{0} \quad p \in \{1, \dots, m\} \setminus \{i, j\} \\ \nabla_p \bar{\Gamma}_i(\mathbf{x}) &= \mathbf{0} \quad p \in \{1, \dots, m\} \setminus \{i\} \\ \nabla_p \bar{\Delta}_i(\mathbf{x}) &= \mathbf{0} \quad p \in \{1, \dots, m\} \setminus \{i\}\end{aligned}\tag{4.12}$$



(a) Exemple d'intersection 2D



(b) Exemple d'intersection 3D

Figure 4.8 – Exemples d'intersection de boîtes englobantes en 2D et 3D.

Les gradients des fonctions Γ_{ij} et Δ_{ij} se rapportant à l'intersection de deux composants i et j s'écrivent :

$$\begin{aligned} \frac{\partial \Gamma_{ij}(\mathbf{x})}{\partial x_{ik}} &= -\frac{\partial \Gamma_{ij}(\mathbf{x})}{\partial x_{jk}} = \begin{cases} 0 & \text{si } \Gamma_{ij}(\mathbf{x}) = 0, \\ \frac{\Gamma_{ij}(\mathbf{x})}{\gamma_{ijk}(\mathbf{x})} \frac{\partial \gamma_{ijk}(\mathbf{x})}{\partial x_{ik}} & \text{sinon.} \end{cases} \\ \frac{\partial \Delta_{ij}(\mathbf{x})}{\partial x_{ik}} &= -\frac{\partial \Delta_{ij}(\mathbf{x})}{\partial x_{jk}} = \begin{cases} 0 & \text{si } \Delta_{ij}(\mathbf{x}) = 0, \\ 0 & \text{si } \delta_{ijk}(\mathbf{x}) \neq \Delta_{ij}(\mathbf{x}), \\ \frac{\partial \delta_{ijk}(\mathbf{x})}{\partial x_{ik}} & \text{sinon.} \end{cases} \end{aligned} \quad (4.13)$$

Les gradients des fonctions $\bar{\Gamma}_i$ et $\bar{\Delta}_i$ se rapportant à l'appartenance du composant i au contenant s'écrivent :

$$\begin{aligned} \frac{\partial \bar{\Gamma}_i(\mathbf{x})}{\partial x_{ik}} &= \begin{cases} 0 & \text{si } \bar{\Gamma}_i(\mathbf{x}) = 0, \\ -\prod_{p=\{1,\dots,d\}\setminus k} \max\{0, \min\{c_p, x_{ip} + b_{ip}/2\} - \max\{0, x_{ip} - b_{ip}/2\}\} \frac{\partial \gamma_{ik}(\mathbf{x})}{\partial x_{ik}} & \text{sinon.} \end{cases} \\ \frac{\partial \bar{\Delta}_i(\mathbf{x})}{\partial x_{ik}} &= \begin{cases} 0 & \text{si } \bar{\Delta}_i(\mathbf{x}) = 0, \\ 0 & \text{si } \bar{\delta}_{ik}(\mathbf{x}) \neq \bar{\Delta}_i(\mathbf{x}), \\ \frac{\partial \bar{\delta}_{ik}(\mathbf{x})}{\partial x_{ik}} & \text{sinon.} \end{cases} \end{aligned} \quad (4.14)$$

4.3.1.3 Construction de fonctions de minimisation

L'ensemble des fonctions présentées précédemment nous permet de construire plusieurs fonctions qualifiant le non-respect des contraintes. La fonction objectif à minimiser sera toujours la somme de deux fonctions : une traduisant les contraintes de non-chevauchement et une autre les contraintes d'appartenance.

$$F(\mathbf{x}) = \omega_{\text{pen}} \sum_{i=1}^{m-1} \sum_{j=i+1}^m f_{ij}(\mathbf{x}) + \omega_{\text{pro}} \sum_{i=1}^m g_i(\mathbf{x}) \quad (4.15)$$

Cinq fonctions ont été construites. Chacune d'elles vérifie les conditions de continuité et différentiabilité. Les spécificités de chacune sont listées ici :

- F_1 : Minimisation de la somme des volumes d'intersection élevés au carré,
- F_2 : Minimisation de la somme des profondeurs de pénétration élevées au carré,
- F_3 : Minimisation de la somme des produits volumes d'intersection / profondeurs de pénétration,
- F_4 : Minimisation de la somme pondérée des profondeurs de pénétration élevées au carré – V1,
- F_5 : Minimisation de la somme pondérée des profondeurs de pénétration élevées au carré – V2.

Les fonctions F_4 et F_5 se différencient par le système de pondération utilisé. L'expression mathématique de chacune de ces fonctions est donnée dans la table 4.1.

Fonction test	$f_{ij}(\mathbf{x})$	$g_i(\mathbf{x})$
F_1	$\Gamma_{ij}^2(\mathbf{x})$	$\bar{\Gamma}_i^2(\mathbf{x})$
F_2	$\Delta_{ij}^2(\mathbf{x})$	$\bar{\Delta}_i^2(\mathbf{x})$
F_3	$\Gamma_{ij}(\mathbf{x}) \Delta_{ij}(\mathbf{x})$	$\bar{\Gamma}_i(\mathbf{x}) \bar{\Delta}_i(\mathbf{x})$
F_4	$\min\{B_i, B_j\} \Delta_{ij}^2(\mathbf{x})$	$B_i \bar{\Delta}_i^2(\mathbf{x})$
F_5	$(B_i + B_j) \Delta_{ij}^2(\mathbf{x})$	$B_i \bar{\Delta}_i^2(\mathbf{x})$

Table 4.1 – Tableau résumant les fonctions de minimisation testées pour le problème de séparation des boîtes.

$$B_i = \prod_{k=1}^d b_{ik}.$$

Le choix de ces fonctions s'explique par plusieurs raisons. Chaque fonction essaie de tirer parti des avantages des fonctions de base décrites précédemment. La fonction F_1 évalue au mieux les contraintes de chevauchement, tout en étant une fonction continue et différentiable. La fonction F_2 n'utilise que les profondeurs de pénétration pour éviter la convergence prématurée dans le cas où des composants sont entièrement situés à l'intérieur d'autres composants. La fonction F_3 est une fonction hybride qui essaie de tirer parti des avantages des fonctions F_1 et F_2 . Enfin, les fonctions F_4 et F_5 utilisent la profondeur de pénétration pondérées deux manières différentes. Le gradient de la fonction F , qui est utilisé par l'algorithme d'optimisation, s'écrit :

$$\nabla F(\mathbf{x}) = \omega_{\text{pen}} \sum_{i=1}^{m-1} \sum_{j=i+1}^m \nabla f_{ij}(\mathbf{x}) + \omega_{\text{pro}} \sum_{i=1}^m \nabla g_i(\mathbf{x}) \quad (4.16)$$

où $\nabla f_{ij}(\mathbf{x}) = (\nabla_1 f_{ij}(\mathbf{x}), \dots, \nabla_m f_{ij}(\mathbf{x}))$ et $\nabla g_i(\mathbf{x}) = (\nabla_1 g_i(\mathbf{x}), \dots, \nabla_m g_i(\mathbf{x}))$. Les gradients des fonctions f_{ij} et g_i sont obtenus à l'aide des gradients présentés ci-avant et de la formule de dérivation d'un produit de fonction. L'algorithme 4.1 présente les différentes étapes de calcul de la fonction test F_1 et de son gradient.

4.3.1.4 Le choix de la fonction à minimiser

Pour déterminer quelle fonction test présente les meilleures caractéristiques de convergence, les différentes fonctions test de minimisation ont été testées sur six exemples, trois en 2D et trois autres en 3D. Les caractéristiques des problèmes générés aléatoirement sont résumées dans la table 4.2.

Problème	Dimension	Nombre de composants (m)	Nombre de ddl (n)	Compacité
1	2	40	80	80.4%
2	2	400	800	95.6%
3	2	1000	2000	95.4%
4	3	30	90	53.0%
5	3	100	300	79.3%
6	3	300	900	70.4%

Table 4.2 – Résumé des caractéristiques des problèmes test pour le problème de séparation des boîtes.

Les poids ω_{pen} et ω_{pro} sont respectivement réglés sur 1 et 10. Ce choix permet de mettre l'accent sur les contraintes d'appartenance. Pour chaque optimisation, un nombre maximum d'itérations de l'algorithme *BFGS* a été fixé à 1000.

Pour comparer l'influence du choix de la fonction à minimiser, un test a été mis en place. Pour chaque fonction test $F_i, \forall i \in \{1, \dots, 5\}$ et à chaque itération de l'optimiseur continu, la matrice solution \mathbf{x} est utilisée pour calculer une fonction qui évalue le non-respect des contraintes de placement. La fonction de comparaison choisie évalue la quantité de chevauchement entre l'ensemble des paires

Algorithme 4.1: Évaluations de la fonction de séparation F et de son gradient ∇F pour le cas test minimisant la somme des carrés des volumes d'intersection (Fonction test F_1).

$(F(\mathbf{x}), \nabla F(\mathbf{x})) \leftarrow \text{compute_separation_function}(m, d, \mathbf{x}, \mathbf{b}, \mathbf{c})$

Données : Nombre de composants m , Dimension du problème d , Matrice des positions des centres des boîtes $\mathbf{x} \in \mathbb{R}^{m \times d}$, Matrice des dimensions des boîtes $\mathbf{b} \in \mathbb{R}^{m \times d}$, Vecteur des dimensions du contenant $\mathbf{c} \in \mathbb{R}^d$

Résultat : Valeur de la fonction $F(\mathbf{x})$ et son gradient $\nabla F(\mathbf{x}) \in \mathbb{R}^{m \times d}$

```

1 Initialisation
2  $F \leftarrow 0$ 
3  $\nabla F \leftarrow \mathbf{0}$ 
4 Évaluation des contraintes de non-chevauchement
5 pour  $i \leftarrow 1$  à  $m-1$  faire
6   pour  $j \leftarrow i+1$  à  $m$  faire
7      $\Gamma_{ij} \leftarrow 1$ 
8     pour  $k \leftarrow 1$  à  $d$  faire
9        $\gamma_{ijk} \leftarrow \max\{0, \min\{x_{ik} + b_{ik}/2, x_{jk} + b_{jk}/2\} - \max\{x_{ik} - b_{ik}/2, x_{jk} - b_{jk}/2\}\}$ 
10       $\Gamma_{ij} \leftarrow \Gamma_{ij} \times \gamma_{ijk}$ 
11    fin
12    si  $\Gamma_{ij} \neq 0$  alors
13      Ajout de la pénalité liée au non-respect de la contrainte de non-chevauchement
14       $F \leftarrow F + \omega_{\text{pen}} \Gamma_{ij}^2$ 
15      Calcul du gradient associé
16       $\nabla_i F \leftarrow \nabla_i F + 2\omega_{\text{pen}} \Gamma_{ij} \nabla_i \Gamma_{ij}$ 
17       $\nabla_j F \leftarrow \nabla_j F - 2\omega_{\text{pen}} \Gamma_{ij} \nabla_i \Gamma_{ij}$ 
18    fin
19  fin
20 fin
21 Évaluation des contraintes d'appartenance
22 pour  $i \leftarrow 1$  à  $m$  faire
23    $\bar{\Gamma}_i \leftarrow 1$ 
24   Calcul du volume de la boîte  $i$  appartenant au contenant
25   pour  $k \leftarrow 1$  à  $d$  faire
26      $\bar{\gamma}_{ik} \leftarrow b_{ik} - \max\{0, \min\{c_k, x_i + b_{ik}/2\} - \max\{0, x_{ik} - b_{ik}/2\}\}$ 
27      $\bar{\Gamma}_i \leftarrow \bar{\Gamma}_i \times \bar{\gamma}_{ik}$ 
28   fin
29   Calcul du volume de la boîte  $i$  à l'extérieur du contenant
30    $\bar{\Gamma}_i \leftarrow \prod_{k=\{1, \dots, d\}} b_{ik} - \bar{\Gamma}_i$ 
31   si  $\bar{\Gamma}_i \neq 0$  alors
32     Ajout de la pénalité liée au non-respect de la contrainte d'appartenance
33      $F \leftarrow F + \omega_{\text{pro}} \bar{\Gamma}_i^2$ 
34     Calcul du gradient associé
35      $\nabla_i F \leftarrow \nabla_i F + 2\omega_{\text{pro}} \bar{\Gamma}_i \nabla_i \bar{\Gamma}_i$ 
36   fin
37 fin

```

de boîtes ainsi que le chevauchement entre le complémentaire du contenant et les boîtes.

$$F_{\text{comp}}(\mathbf{x}) = \sum_{i=1}^{m-1} \sum_{j=i+1}^m \Gamma_{ij}(\mathbf{x}) + \sum_{i=1}^m \bar{\Gamma}_i(\mathbf{x}) \quad (4.17)$$

Cette fonction est continue par morceaux et n'est pas différentiable pour tout point \mathbf{x} , raison pour laquelle elle n'a pas été retenue comme fonction test. Les figures 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 montrent les solutions obtenues par chaque optimisation pour les différents problèmes présentés ainsi que les résultats des évaluations de la fonction de comparaison F_{comp} . Pour chaque problème, les solutions initiales sont identiques : toutes les courbes de convergence ont le même point d'origine. Pour chaque fonction test $F_i, \forall i \in \{1, \dots, 5\}$, chaque nouvelle itération de l'optimiseur permet de faire diminuer la valeur de F_i . Les courbes de convergence sont donc des courbes strictement décroissantes. En revanche, on peut observer sur les courbes montrant l'évolution de la fonction F_{comp} pour les différents problèmes et optimisations, qu'elles ne sont pas strictement décroissantes.

Sur les six exemples réalisés, les fonctions F_2 , F_4 , et F_5 n'ont jamais obtenu le meilleur résultat. La fonction F_1 a obtenu le meilleur résultat sur le dernier problème, la fonction F_3 a obtenu tous les autres meilleurs résultats. En terme de vitesse de convergence, la fonction F_3 apparaît clairement la plus rapide. La table 4.3 résume numériquement les valeurs de F_{comp} obtenues par les différentes fonctions test pour les différents problèmes.

Problème	F_1	F_2	F_3	F_4	F_5
1	1.0	1.7	0	5.1	5.1
2	6.5	115.6	3.8	117.3	102.2
3	17.5	213.6	8.8	337.6	186.5
4	15.9	51.3	1.9×10^{-6}	128.0	50.5
5	29.4	344.7	17.8	665.4	264.5
6	113.8	958.7	140.4	186.4	803.5

Table 4.3 – Valeurs de la fonction de comparaison F_{comp} pour les solutions aux différents problèmes avec les différentes fonctions test. Les cellules grisées représentent les meilleures solutions identifiées pour chaque problème test.

Pour quantifier l'ensemble des déplacements réalisés par l'algorithme de séparation, une distance est introduite. Elle évalue en moyenne la distance parcourue par les composants entre leur position initiale (\mathbf{x}^{ori}) et leur position optimisée ou finale (\mathbf{x}^{opt}). Cette distance est normée par rapport à la plus grande dimension du contenant, ce qui permet d'obtenir une distance variant entre zéro et un. Elle s'écrit :

$$\mathcal{D}(\mathbf{x}^{\text{ori}}, \mathbf{x}^{\text{opt}}) = \frac{1}{m \max_{k=\{1, \dots, d\}} \{c_k\}} \sum_{i=1}^m \sqrt{\sum_{k=1}^d (x_{ik}^{\text{opt}} - x_{ik}^{\text{ori}})^2} \quad (4.18)$$

La table 4.4 présente les distances entre solutions initiales et finales pour chaque problème test et chaque fonction objectif. Les fonctions test qui ont obtenu les valeurs minimales pour F_{comp} ne sont pas forcément celles qui ont déplacé le plus les composants.

Problème	F_1	F_2	F_3	F_4	F_5
1	0.1026	0.0892	0.0954	0.0848	0.0862
2	0.0506	0.0415	0.0441	0.0446	0.0422
3	0.0329	0.0319	0.0328	0.0316	0.0299
4	0.1851	0.1762	0.1752	0.1763	0.1771
5	0.1567	0.1170	0.1650	0.1174	0.1169
6	0.1170	0.0940	0.1228	0.0929	0.0928

Table 4.4 – Distances entre solutions initiales et finales pour chaque problème test et chaque fonction objectif. La distance utilisée est présentée équation 4.18. Les cellules grisées représentent les distances pour lesquelles la meilleure solution a été trouvée.

En conclusion, le choix de la fonction à minimiser influence les résultats de l'algorithme de séparation et joue sur sa vitesse de convergence. Le choix de la fonction F_3 , somme des produits des volumes d'intersection et profondeurs de pénétration, s'avère être un choix intéressant. Cette fonction présente des résultats meilleurs que les autres fonctions test avec une convergence plus rapide. De plus,

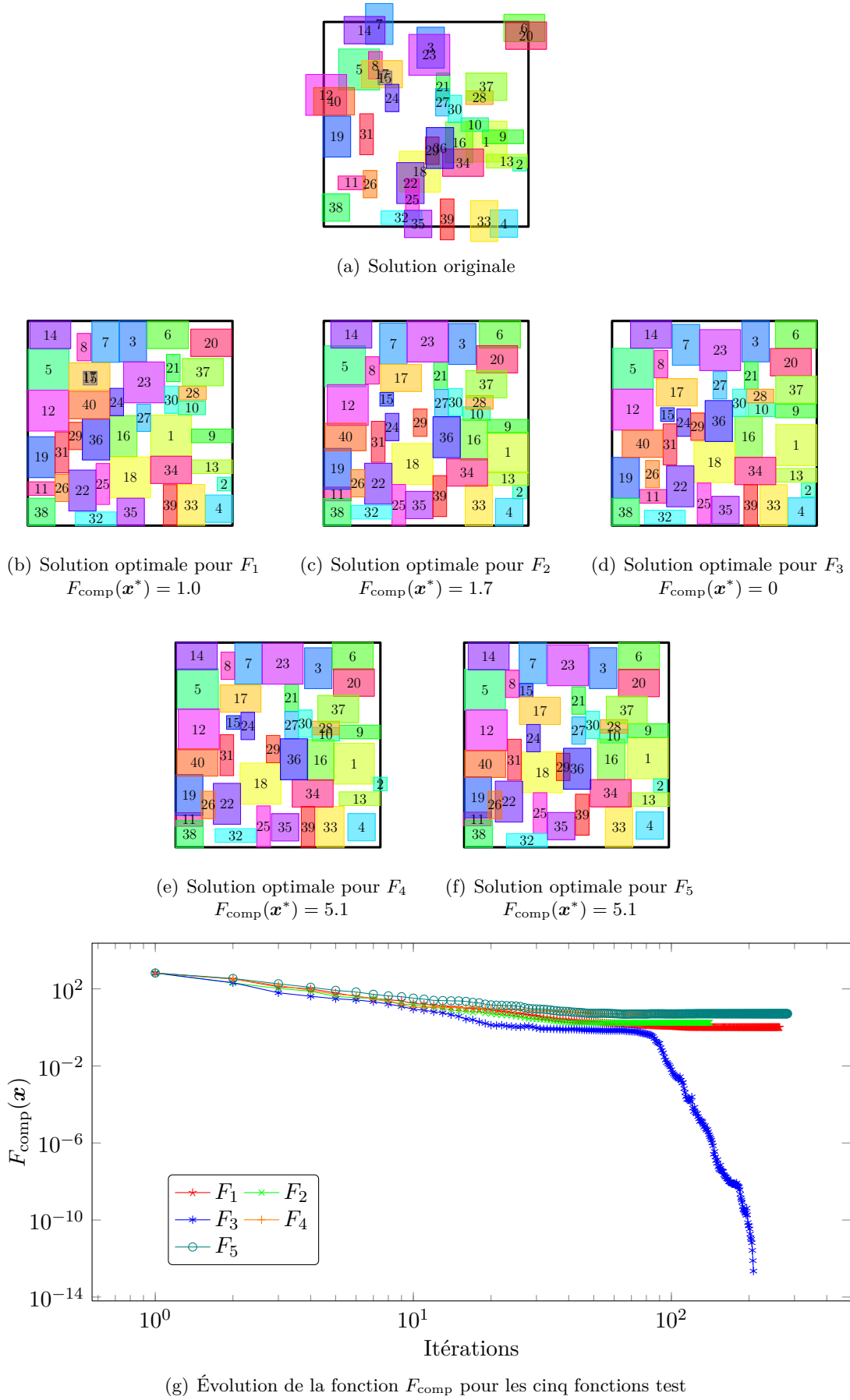


Figure 4.9 – Influence du choix de la fonction à minimiser pour le problème de séparation 2D avec 40 rectangles. La compacité du problème est de 80.4%. La convergence de chaque cas est donnée sur la sous-figure (g).

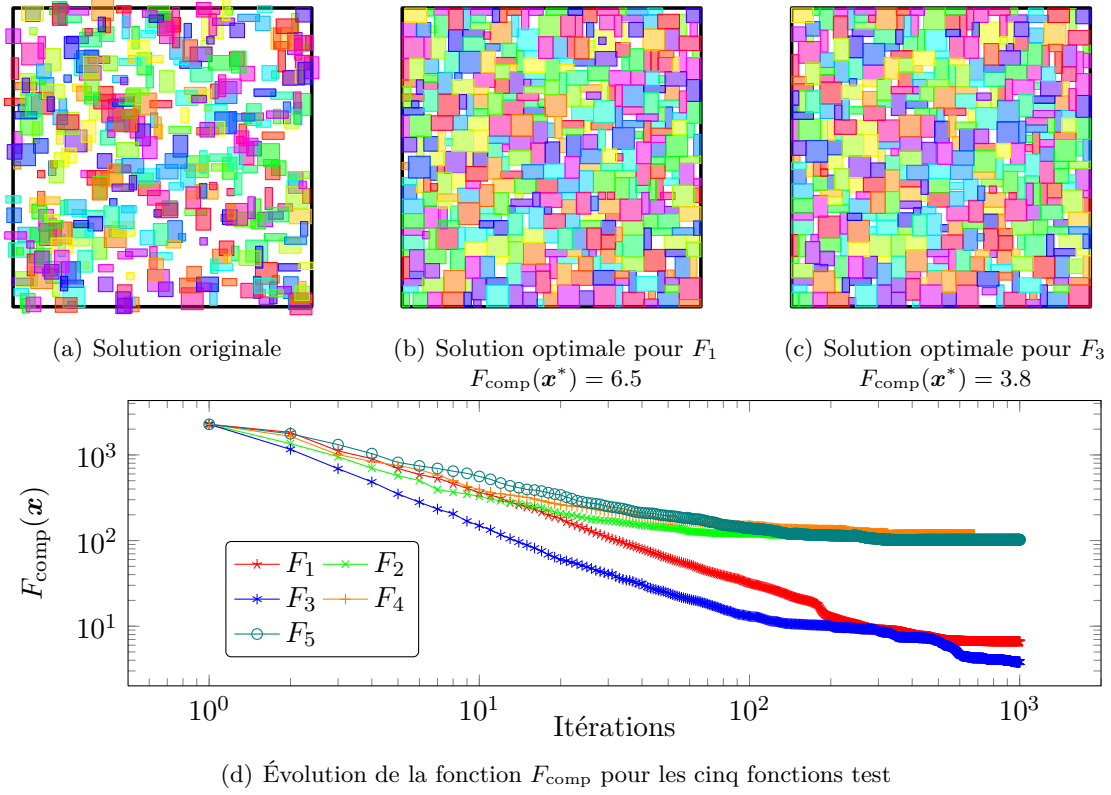


Figure 4.10 – Influence du choix de la fonction à minimiser pour le problème de séparation 2D avec 400 rectangles. La compacité du problème est de 95.6%. Seules les deux meilleures solutions sont représentées. La table 4.3 donne les valeurs de la fonction F_{comp} pour les différentes fonctions test.

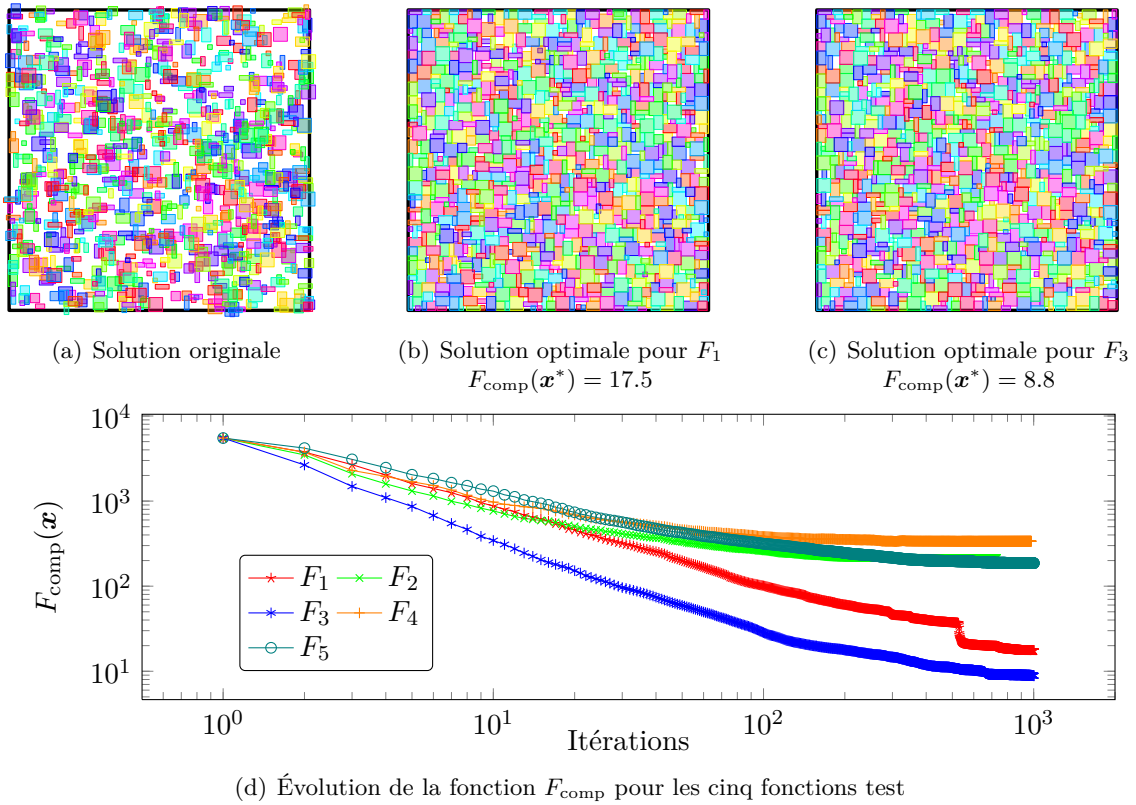


Figure 4.11 – Influence du choix de la fonction à minimiser pour le problème de séparation 2D avec 1000 rectangles. La compacité du problème est de 95.4%. Seules les deux meilleures solutions sont représentées. La table 4.3 donne les valeurs de la fonction F_{comp} pour les différentes fonctions test.

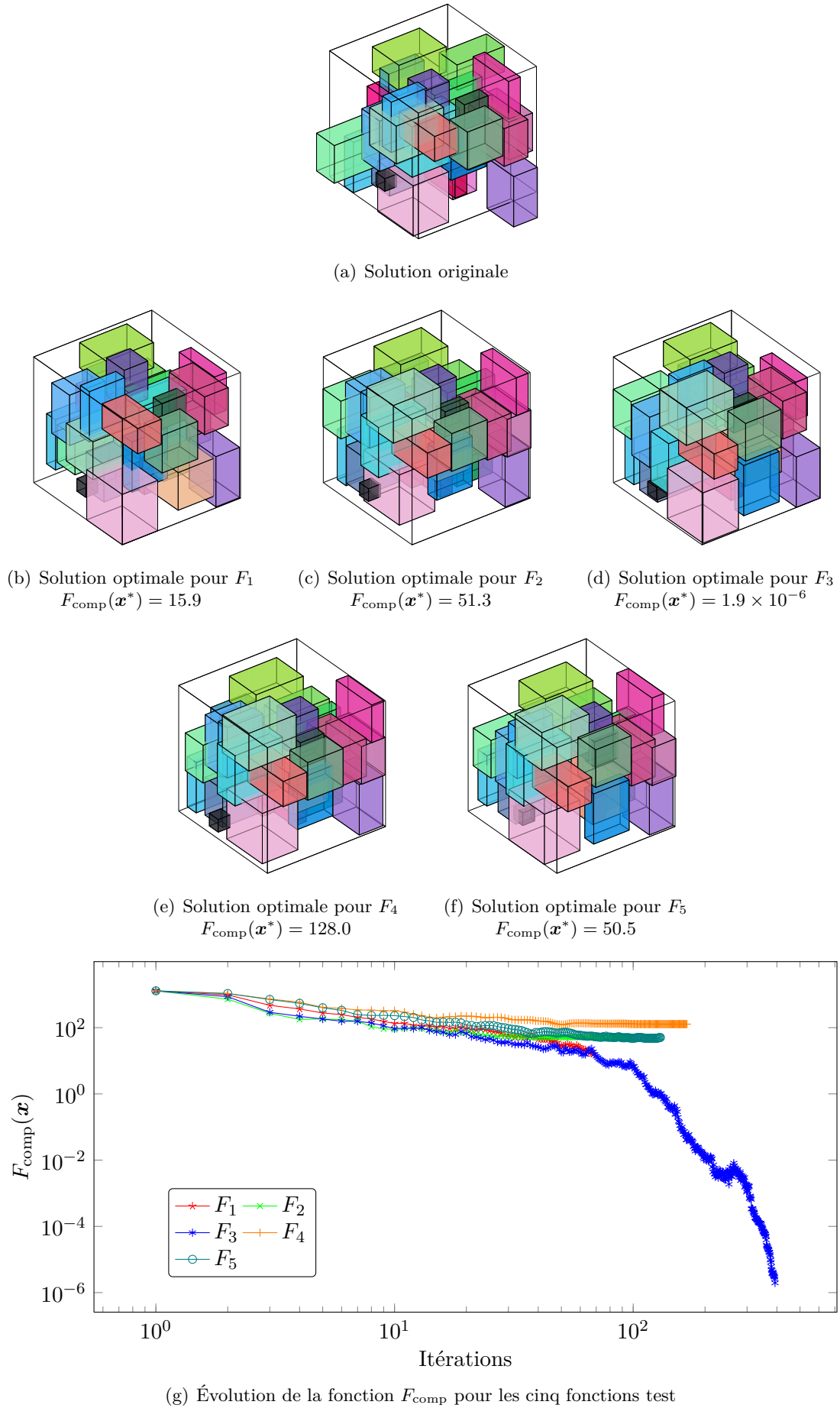


Figure 4.12 – Influence du choix de la fonction à minimiser pour le problème de séparation 3D avec 30 parallélépipèdes. La compacité du problème est de 53%.

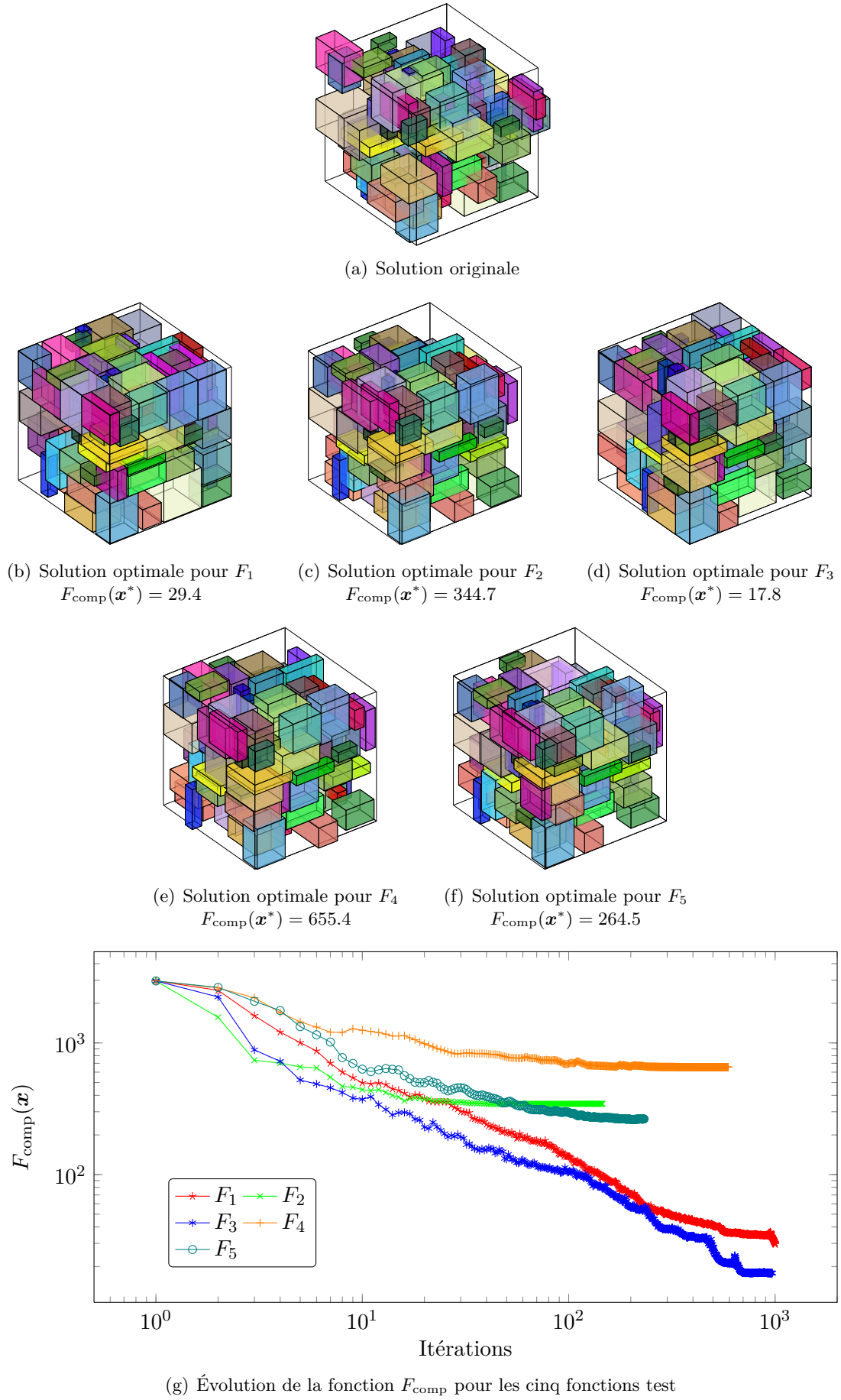


Figure 4.13 – Influence du choix de la fonction à minimiser pour le problème de séparation 3D avec 100 parallélépipèdes. La compacité du problème est de 79.3%.

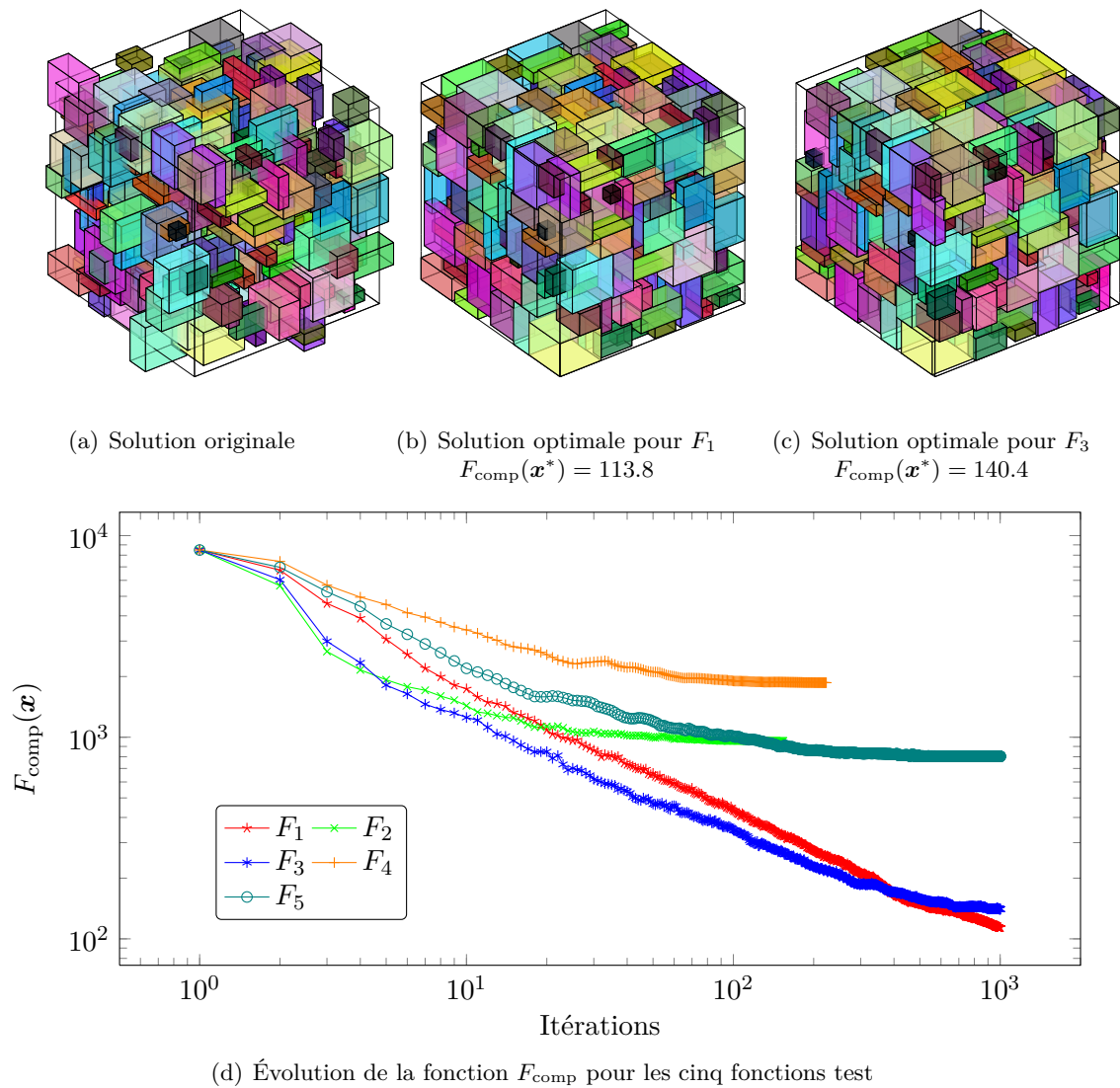


Figure 4.14 – Influence du choix de la fonction à minimiser pour le problème de séparation 3D avec 300 parallélépipèdes. La compacité du problème est de 70.4%. Seules les deux meilleures solutions sont représentées. La table 4.3 donne les valeurs de la fonction F_{comp} pour les différentes fonctions test.

elle permet de séparer des composants se chevauchant intégralement, contrairement la fonction test F_1 . En effet, le gradient de la fonction F_1 est nul lorsque cette situation se produit, causant un arrêt prématuré de l'optimisation. Le premier exemple 2D avec 40 composants présente une telle situation avec les composants numérotés 15 et 17 (Figure 4.9(b)).

4.3.1.5 Prise en compte des orientations discrètes des parallélépipèdes

Lorsqu'on souhaite optimiser le placement d'un ensemble de boîtes, l'orientation de ces dernières peut être une variable d'optimisation. Pour conserver le cadre de travail présenté ci-avant, les rotations sont orthogonales, de manière à toujours avoir des boîtes alignées sur le système d'axe.

Le choix a été fait de piloter le centre des boîtes. Cela permet de gérer les rotations de la boîte par rapport à son centre de volume plus aisément. Dans ce cas, les rotations consistent à effectuer des permutations sur les dimensions des boîtes. En 2D, chaque boîte peut avoir deux orientations. Une rotation est donc prise en compte en intervertissant les deux dimensions d'un rectangle. En 3D, chaque boîte peut avoir $3!$ orientations soit six au total. Les six orientations d'une boîte sont présentées figure 4.15.

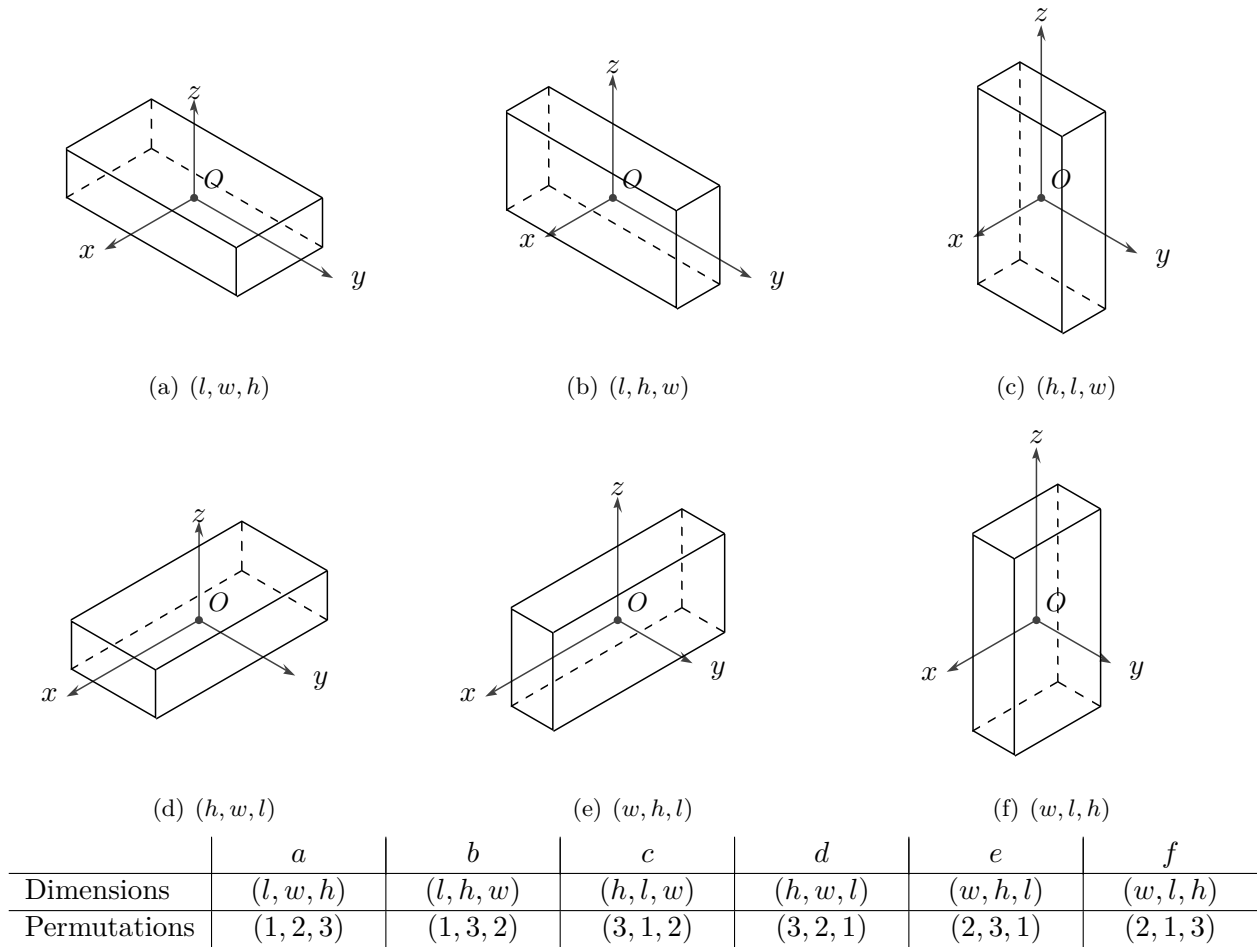


Figure 4.15 – Présentation des six orientations orthogonales possibles d'un parallélépipède de dimensions (l, w, h) dans le repère $(O, \vec{x}, \vec{y}, \vec{z})$. Image empruntée à Tiwari *et al.* [TFF08].

Enfin, pour chacune de ces orientations, une boîte peut encore avoir quatre orientations de faces différentes présentées figure 4.16. Ces deux types d'orientations sont représentés à l'aide de nombres discrets dans l'algorithme d'optimisation global. Dans les problèmes C&P, l'orientation des faces n'est pas un critère, seul le premier type d'orientation est utilisé. En revanche, les problèmes d'agencement peuvent utiliser les deux orientations.

4.3.1.6 Extensions futures

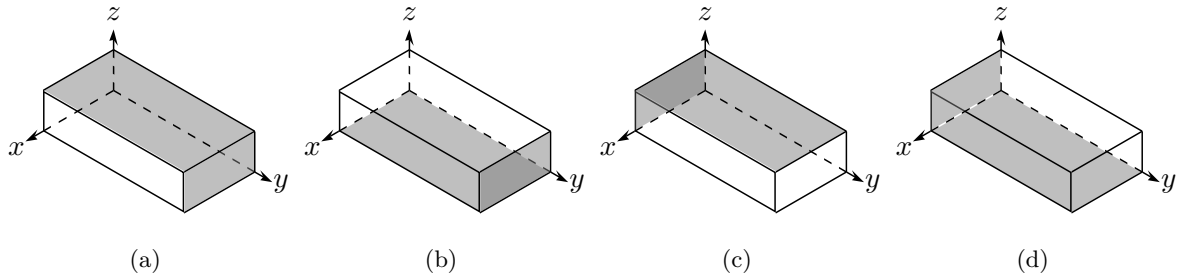


Figure 4.16 – Présentation des quatre orientations possibles pour une même boîte englobante. Image empruntée à Tiwari *et al.* [TFF08].

La méthode présentée ici considèrerait que l'ensemble des boîtes possédait les mêmes degrés de liberté et pouvait occuper n'importe quelle position à l'intérieur du contenant. Lors de la résolution des problèmes d'agencement, ceci n'est plus forcément vrai : des contraintes propres au problème à résoudre peuvent venir s'ajouter aux contraintes de placement classiques. Certaines de ces contraintes peuvent être prises en compte dans l'algorithme de séparation :

- certains composants peuvent être contraints de se déplacer selon des directions privilégiées de l'espace. Dans le cas où ces directions correspondent aux directions du système d'axe, il suffit de supprimer certains degrés de liberté. Dans le cas où les directions sont quelconques, il faut utiliser une représentation paramétrique de la position des composants à placer.
- certains composants peuvent être contraints de se situer dans une zone de l'espace. Pour modéliser ce cas de figure, il faut considérer que ces composants ne doivent plus être contenus dans le contenant, mais dans une zone définie par l'utilisateur.
- un espace d'accessibilité peut être pris en compte. Il suffit pour cela de faire grossir virtuellement certaines faces des composants et considérer des composants virtuels, qui n'intersectent qu'un sous-ensemble des composants.

Ces extensions peuvent être prises en compte assez facilement dans l'algorithme, le défi consiste à mettre en place les structures de données correspondantes et à saisir les données spécifiques de l'utilisateur.

Enfin, l'algorithme présenté ici peut directement être utilisé pour la résolution de problème d'agencement de puces électroniques. Dans ce genre de problèmes, il est habituel de calculer une solution initiale à l'aide d'une optimisation continue minimisant la longueur des connexions entre les composants, mais ne prenant pas en compte les contraintes de placement. Cette optimisation fournit une solution optimale en terme de longueur de connexion, mais où les composants se chevauchent et ne respectent pas forcément la contrainte d'appartenance. Notre algorithme peut être utilisé pour rendre admissible la solution, et être comparé aux méthodes de légalisation utilisées [Ege03].

Cette section a présenté une variante de l'algorithme de séparation. Toutefois, les problèmes de placement ne se limitent pas aux géométries rectangulaires et parallélépipédiques. Les sections suivantes présentent des algorithmes de séparation pour des composants de géométries complexes.

4.3.2 Algorithme de séparation en translation pour polygones

L'algorithme de séparation présenté ici est adapté aux composants représentés sous forme de polygones à orientations discrètes. Il a été présenté par Imamichi *et al.* [IYN09] dans le cadre de la résolution d'un problème de découpe de composants de géométrie irrégulière. Cet article nous a inspiré les autres algorithmes de séparation présentés dans ce mémoire.

Lorsque les composants ne sont plus de géométrie simple, il devient difficile et coûteux de calculer l'aire de chevauchement entre chaque paire de composants pour évaluer le non-respect des contraintes de placement. Dans la section précédente, nous avons vu qu'il existait d'autres moyens pour quantifier le chevauchement de deux composants, notamment avec la profondeur de pénétration δ . L'algorithme présenté ici base sa fonction de minimisation sur les profondeurs de pénétration, et utilise pour cela les polygones de non-recouvrement et d'appartenance.

Les contraintes de non-chevauchement et d'appartenance sont évaluées dans une seule et même

fonction $F(\mathbf{x}, \mathbf{o})$, où \mathbf{x} et \mathbf{o} représentent respectivement la matrice de position des composants et le vecteur d'orientation des composants. L'orientation $\theta_i \in \mathbf{o}$ de chaque composant C_i est choisie parmi un ensemble discret d'orientations Q_i défini par l'utilisateur. Le problème de séparation s'écrit :

$$(\text{Sep_}2D) \left\{ \begin{array}{l} \min F(\mathbf{x}, \mathbf{o}) = \omega_{\text{pen}} \sum_{i=1}^{m-1} \sum_{j=i+1}^m f_{ij}(\mathbf{x}, \mathbf{o}) + \omega_{\text{pro}} \sum_{i=1}^m g_i(\mathbf{x}, \mathbf{o}) \\ \text{s.c.} \quad \left| \begin{array}{l} \mathbf{x} \in \mathbb{R}^{m \times 2} \\ \mathbf{o} \in \mathbf{Q} \end{array} \right. \end{array} \right. \quad (4.19)$$

avec $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)$, $\mathbf{o} = (\theta_1, \theta_2, \dots, \theta_m)$, $\mathbf{Q} = Q_1 \times Q_2 \times \dots \times Q_m$. Les fonctions f_{ij} et g_i évaluent respectivement le non-respect de la contrainte de non-chevauchement entre les composants i et j , ainsi que le non-respect de la contrainte d'appartenance entre le composant i et le contenant. Si ces deux contraintes sont satisfaites, alors les valeurs de f_{ij} et g_i sont nulles.

Pour simplifier le problème, seules les translations des différents composants sont autorisées pour minimiser la fonction F . La prise en compte d'orientations discrètes, qui ne sont pas limitées aux orientations orthogonales, se fait au niveau de l'optimisation globale. En considérant des poids ω_{pen} et ω_{pro} égaux, le problème de minimisation se réécrit alors :

$$(\text{Sep_}2D') \left\{ \begin{array}{l} \min_{\mathbf{x} \in \mathbb{R}^{m \times 2}} F(\mathbf{x}) = \sum_{i=1}^{m-1} \sum_{j=i+1}^m f_{ij}(\mathbf{x}) + \sum_{i=1}^m g_i(\mathbf{x}) \end{array} \right. \quad (4.20)$$

Les intersections entre composants sont évaluées en utilisant les profondeurs de pénétration entre composants, pour lesquelles les gradients analytiques peuvent être facilement évalués. La profondeur de pénétration $\delta(P_i, P_j)$ entre deux polygones P_i et P_j [AGHP⁺00] est définie par :

$$\delta(P_i, P_j) = \min \left\{ \|z\| \mid \text{int}(P_i) \cap (P_j \oplus z) = \emptyset, z \in \mathbb{R}^2 \right\} \quad (4.21)$$

où $\text{int}(P_i)$ désigne l'intérieur du polygone P_i , et $(P_j \oplus z)$ représente le polygone P_j translaté du vecteur z . $\delta(P_i, P_j)$ correspond à la distance de translation minimale pour assurer la séparation des polygones P_i et P_j . L'évaluation de la profondeur de pénétration $\delta(P_i, P_j)$ se fait à l'aide du polygone de non-recouvrement de P_i et P_j . La définition mathématique et les méthodes de calcul de ce polygone sont présentées en annexe section B.4.1. La fonction f_{ij} s'exprime à l'aide de la profondeur de pénétration des polygones P_i et P_j .

$$f_{ij}(\mathbf{x}) = \delta(P_i \oplus \mathbf{x}_i, P_j \oplus \mathbf{x}_j)^r, 1 \leq i < j \leq m \quad (4.22)$$

avec r est un paramètre positif. Ce paramètre sera réglé de manière à s'assurer de la différentiabilité de f_{ij} . Le vecteur gradient de f_{ij} se calcule de la manière suivante :

$$\begin{aligned} f_{ij}(\mathbf{x}) &= \|z\|^r \\ \nabla_i f_{ij}(\mathbf{x}) &= -\nabla_j f_{ij}(\mathbf{x}) = r \|z\|^{r-2} z \\ \nabla_k f_{ij}(\mathbf{x}) &= \mathbf{0}, k \in \{1, \dots, m\} \setminus \{i, j\} \end{aligned} \quad (4.23)$$

où z désigne le vecteur de translation minimale tel que les polygones P_i et P_j ne se chevauchent plus ($z \in \mathbb{R}^2$).

La figure 4.17 présente le calcul des profondeurs de pénétration pour différents points \mathbf{p}_k . Le polygone de cette figure représente le polygone de non-recouvrement de deux composants C_i et C_j . Les points \mathbf{p}_k représentent la position relative des composants C_i et C_j : $\mathbf{p}_k = \mathbf{x}_j - \mathbf{x}_i$. Se trouvant tous à l'intérieur du polygone, les points \mathbf{p}_k indiquent une collision entre les composants C_i et C_j . Pour chacun de ces points, le point de la frontière le plus proche est calculé. Les vecteurs formés entre les points \mathbf{p}_k et leurs points les plus proches fournissent des directions de séparation, et les normes de ces vecteurs correspondent aux profondeurs de pénétration δ . Pour les points situés sur l'axe médian du polygone, il n'y a pas unicité de la direction de séparation. Dans ces cas de figure, la direction de séparation est choisie aléatoirement parmi les différentes possibilités.

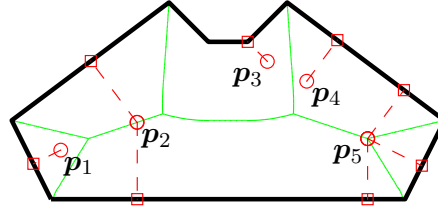


Figure 4.17 – Calcul des profondeurs de pénétration en 2D. Pour chaque point $p_k, \forall k \in \{1, \dots, 5\}$, le point du polygone le plus proche est calculé. La distance entre chaque point p_k et son point le plus proche sur le polygone correspond à la profondeur de pénétration entre le point et le polygone. Pour les points p_2 et p_5 situés sur l'axe médian représenté en trait plein, il n'y a pas unicité du point le plus proche.

La contrainte d'appartenance des composants au contenant s'écrit :

$$g_i(\mathbf{x}) = \delta \left(\text{cl}(\bar{C}), P_i \oplus \mathbf{x}_i \right)^r, 1 \leq i \leq m \quad (4.24)$$

où $\text{cl}(\bar{C})$ désigne l'adhérence¹ du complémentaire de C . Les calculs de la fonction g_i sont basés sur l'utilisation sur les polygones d'appartenance. La définition mathématique et les méthodes de calcul de ces polygones sont présentées en annexe section B.4.2. La figure 4.18 illustre le calcul de la fonction g_i pour le polygone d'appartenance du polygone P_i . Les différents points p_k représentent les points de référence des polygones situés à l'extérieur du polygone d'appartenance.

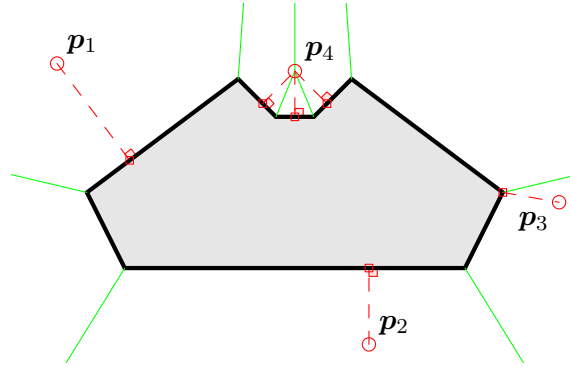


Figure 4.18 – Calcul des contraintes d'appartenance g_i pour différents points p_k situés à l'extérieur du polygone. Pour certains points situés sur l'axe médian, plusieurs points du polygone peuvent donner la même profondeur de pénétration : le point p_4 a trois possibilités pour réintégrer le polygone d'appartenance.

Les gradients de la fonction g_i s'écrivent :

$$\begin{aligned} g_i(\mathbf{x}) &= \|\mathbf{z}\|^r \\ \nabla_i g_i(\mathbf{x}) &= r \|\mathbf{z}\|^{r-2} \mathbf{z} \\ \nabla_k g_i(\mathbf{x}) &= \mathbf{0}, \quad k \in \{1, \dots, m\} \setminus \{i\} \end{aligned} \quad (4.25)$$

où \mathbf{z} représente le vecteur de translation minimale pour que le polygone P_i appartienne au contenant. La figure 4.19 montre l'évolution de la fonction f_{ij} lors de l'entrée en collision des polygones P_i et P_j pour différentes valeurs du paramètre r . Le trait pointillé est utilisé pour matérialiser le point de contact entre les deux polygones. Pour s'assurer de la différentiabilité des fonctions f_{ij} et g_i lors d'entrées en collision, la valeur de r est choisie égale à 2. Les gradients de ces fonctions se réécrivent :

$$\begin{aligned} \nabla_i f_{ij}(\mathbf{x}) &= -\nabla_j f_{ij}(\mathbf{x}) = 2\mathbf{z} \\ \nabla_i g_i(\mathbf{x}) &= 2\mathbf{z} \end{aligned} \quad (4.26)$$

où \mathbf{z} représente le vecteur de translation minimale qui assurera la séparation de P_i et P_j , ou l'appartenance de P_i au contenant.

1. L'adhérence (*closure* en anglais) d'une partie X d'un espace topologique E est le plus petit ensemble fermé de E qui contienne X

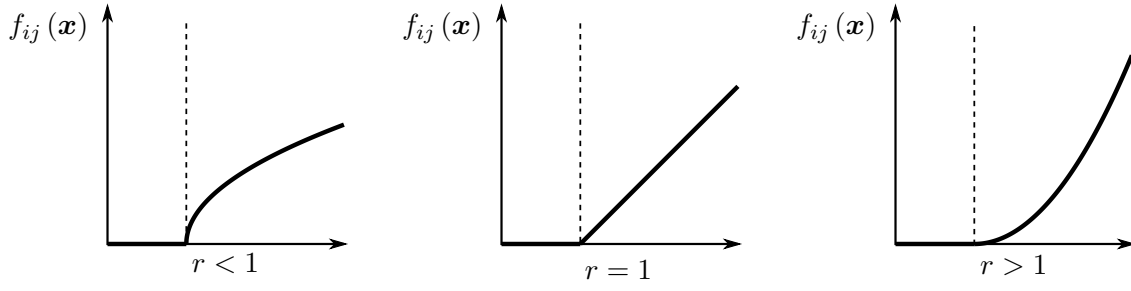


Figure 4.19 – Évolution de la fonction $f_{ij}(\mathbf{x})$ lors de l'entrée en collision des composants C_i et C_j pour différentes valeurs de r . Image empruntée à Imamichi *et al.* [IYN09].

Les calculs des fonctions f_{ij} et g_i sont donc basés sur l'utilisation des polygones de non-recouvrement et d'appartenance, un algorithme de détection d'appartenance de points à des polygones et une méthode de calcul de point le plus proche entre un point et un ensemble de segments. Les étapes de calcul de la fonction de séparation F et de son gradient sont présentées dans l'algorithme 4.2.

Algorithme 4.2: Évaluations de la fonction de séparation F et de son gradient ∇F pour le problème de séparation en translation de polygones.

$(F(\mathbf{x}), \nabla F(\mathbf{x})) \leftarrow \text{compute_separation_function}(m, \mathbf{x}, \mathbf{NFP}, \mathbf{IFP})$

Données : Nombre de composants m , Matrice position $\mathbf{x} \in \mathbb{R}^{m \times 2}$, Liste des \mathbf{NFP} et \mathbf{IFP}

Résultat : Valeur de la fonction $F(\mathbf{x})$ et son gradient $\nabla F(\mathbf{x}) \in \mathbb{R}^{m \times 2}$

```

1  Initialisation
2   $F \leftarrow 0$ 
3   $\nabla F \leftarrow \mathbf{0}$ 
4  Évaluation des contraintes de non-chevauchement
5  pour  $i \leftarrow 1$  à  $m - 1$  faire
6      pour  $j \leftarrow i + 1$  à  $m$  faire
7          si  $(\mathbf{x}_j - \mathbf{x}_i) \in \mathbf{NFP}(P_i, P_j)$  alors
8              Calculer le point  $\mathbf{p}$  de  $\mathbf{NFP}(P_i, P_j)$  le proche de  $(\mathbf{x}_j - \mathbf{x}_i)$ 
9               $\mathbf{z} \leftarrow (\mathbf{x}_j - \mathbf{x}_i) - \mathbf{p}$ 
10              $F \leftarrow F + \omega_{\text{pen}} |\mathbf{z}|^2$ 
11              $\nabla_i F \leftarrow \nabla_i F + 2\omega_{\text{pen}} \mathbf{z}$ 
12              $\nabla_j F \leftarrow \nabla_j F - 2\omega_{\text{pen}} \mathbf{z}$ 
13         fin
14     fin
15 fin
16 Évaluation des contraintes d'appartenance
17 pour  $i \leftarrow 1$  à  $m$  faire
18     si  $\mathbf{x}_i \notin \mathbf{IFP}(P_i)$  alors
19         Calculer le point  $\mathbf{p}$  de  $\mathbf{IFP}(P_i)$  le proche de  $\mathbf{x}_i$ 
20          $\mathbf{z} \leftarrow \mathbf{x}_i - \mathbf{p}$ 
21          $F \leftarrow F + \omega_{\text{pro}} |\mathbf{z}|^2$ 
22          $\nabla_i F \leftarrow \nabla_i F + 2\omega_{\text{pro}} \mathbf{z}$ 
23     fin
24 fin
```

La figure 4.20 illustre l'évolution de la convergence du problème de séparation avec la méthode *BFGS* sur l'instance de découpe de formes irrégulières *Dighe 2* [DJ96]. L'évolution numérique de la fonction au cours des itérations est présentée sur la sous-figure (m).

Cette version de l'algorithme de séparation est très bien adaptée aux problèmes avec des composants à orientations discrètes. Toutefois, elle ne permet pas de prendre en compte les composants avec des orientations continues. L'adaptation de cet algorithme au cas 3D est aussi difficile. Les volumes de

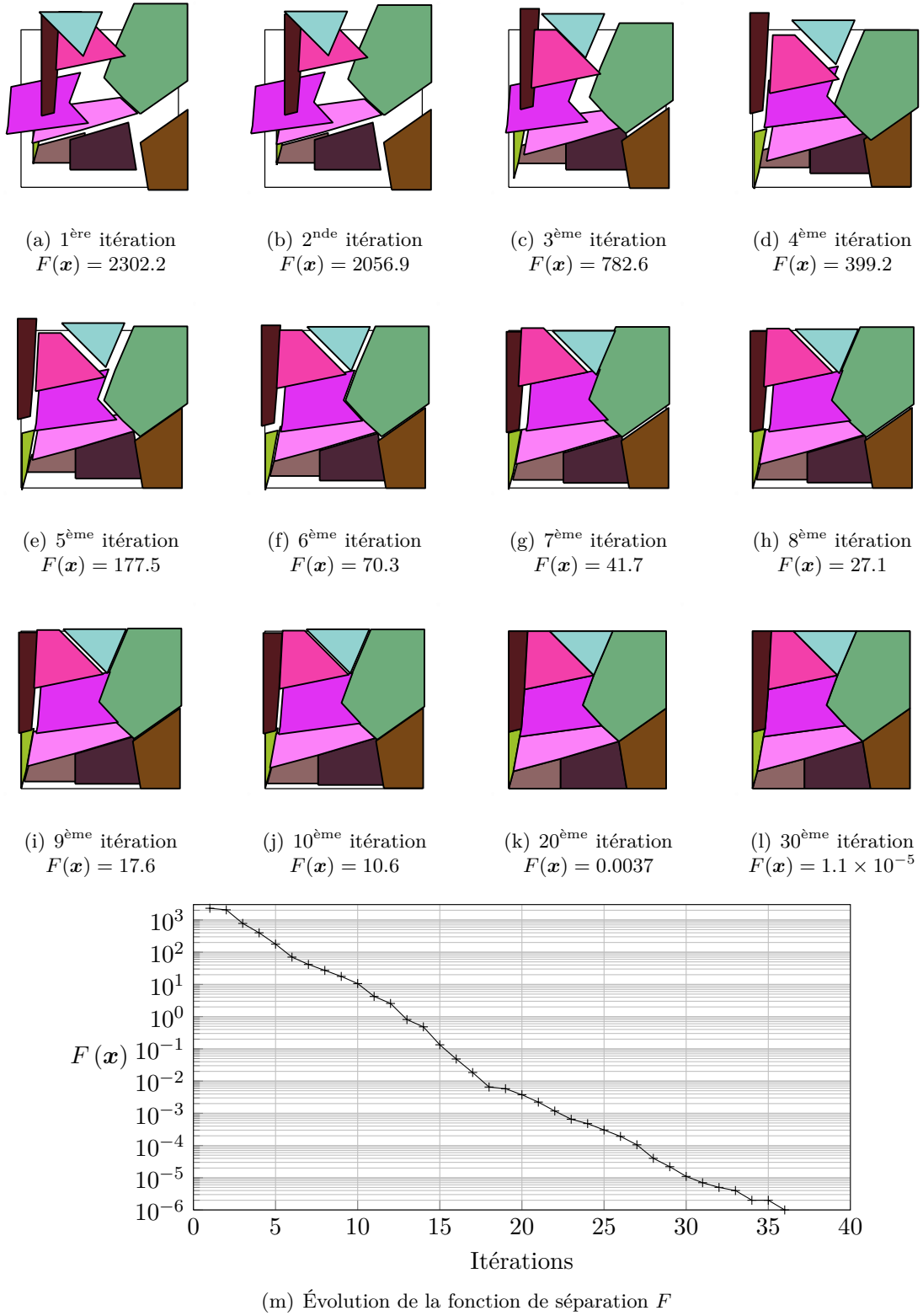


Figure 4.20 – Convergence de l’algorithme de séparation *ILSQN* sur l’instance *Dighe 2* [DJ96]. La sous-figure (m) montre l’évolution de la fonction F .

non-recouvrement entre composants et ainsi que les volumes d’appartenance, adaptations respectives des polygones de non-recouvrement et des polygones d’appartenance, ne sont pas facilement calculables. En effet, le calcul des sommes de Minkowski 3D est complexe même dans le cas de polyèdres (polyèdres fermés convexes) [FH05]. Ces différents éléments nous conduisent à étudier une autre variante de l’algorithme de séparation basée sur des assemblages de primitives, à savoir des cercles et des sphères.

4.3.3 Algorithme de séparation pour des composants modélisés par un assemblage de cercles / sphères

L'algorithme précédent présente l'avantage de travailler avec la vraie géométrie des composants. Son inconvénient est qu'il ne permet pas de modéliser la libre rotation des composants lors de la séparation. L'utilisation d'une représentation des composants sous la forme d'assemblages de sphères présente plusieurs avantages. De plus, elle permet de passer facilement du cas 2D au cas 3D en modélisant la libre rotation des composants.

Imamichi *et al.* [IN07, IAG⁺08] ont proposé un programme d'optimisation pour séparer des composants modélisés par des assemblages de sphères se chevauchant. Cette section reprend les principes de base permettant d'écrire le programme de séparation et étend l'algorithme au contenant de géométrie complexe en 2D et 3D. Enfin, nous précisons comment transformer la géométrie des composants en assemblages de cercles / sphères.

Soit un ensemble de m composants $\mathcal{C} = \{C_1, \dots, C_m\}$. Chaque composant C_i est composé de n_i sphères $\{S_{i1}, S_{i2}, \dots, S_{in_i}\}$. Soient \mathbf{c}_{ij} le vecteur représentant la position du centre de la sphère S_{ij} et r_{ij} le rayon de la sphère S_{ij} ($i = 1, \dots, m; j = 1, \dots, n_i$). À chaque composant C_i est associé un point de référence \mathbf{r}_i , qui permettra de placer et d'orienter le composant. Le programme d'optimisation pilotera l'ensemble des points de référence \mathbf{r}_i . Pour simplifier les écritures, chaque composant est initialement positionné de telle manière que son point de référence soit situé à l'origine du repère. Ce point de référence sera choisi comme le centre de la sphère englobante du composant.

Les positions relatives des sphères du composant C_i sont stockées dans une matrice \mathbf{P}_i . Cette matrice comporte n_i lignes et d colonnes. Chaque ligne correspond à la position de la sphère j par rapport au point de référence du composant C_i . $p_{ij,x}$ désigne donc la position selon l'axe x du centre de la sphère j du composant C_i .

L'algorithme de séparation est basé sur la minimisation de la somme des profondeurs de pénétration entre les différents éléments du problème :

$$(\text{Sep}) \begin{cases} \min & F(\mathbf{v}) = \omega_{\text{pen}} F_{\text{pen}}(\mathbf{v}) + \omega_{\text{pro}} F_{\text{pro}}(\mathbf{v}) \\ \text{s.c.} & \mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_m) \in \mathbb{R}^{6m} \end{cases} \quad (4.27)$$

avec

$$\begin{aligned} F_{\text{pen}}(\mathbf{v}) &= \sum_{1 \leq i < k \leq m} \sum_{j=1}^{n_i} \sum_{l=1}^{n_k} f_{ijkl}^{\text{pen}}(\mathbf{v}) \\ F_{\text{pro}}(\mathbf{v}) &= \sum_{i=1}^m \sum_{j=1}^{n_i} f_{ij}^{\text{pro}}(\mathbf{v}) \end{aligned} \quad (4.28)$$

où f_{ijkl}^{pen} désigne la profondeur de pénétration au carré de la sphère S_{ij} avec la sphère S_{kl} . f_{ij}^{pro} désigne la profondeur de pénétration au carré de la sphère S_{ij} avec le contenant. L'utilisation du carré permet de s'assurer de la différentiabilité de la fonction F .

$$f_{ijkl}^{\text{pen}}(\mathbf{v}) = (\delta(S_{ij}(\mathbf{v}_i), S_{kl}(\mathbf{v}_k)))^2 \quad (4.29a)$$

$$f_{ij}^{\text{pro}}(\mathbf{v}) = \left(\delta(S_{ij}(\mathbf{v}_i), \text{cl}(\overline{C})) \right)^2 \quad (4.29b)$$

avec \overline{C} le complémentaire de C et $\text{cl}(\overline{C})$ l'adhérence du complémentaire de C . La profondeur de pénétration entre deux composants C_1 et C_2 est définie comme la distance de translation minimale permettant de séparer C_1 et C_2 . Dans le cas où C_1 et C_2 sont des sphères de rayons respectifs r_1 et r_2 et dont les centres respectifs sont donnés par les vecteurs positions \mathbf{c}_1 et \mathbf{c}_2 , la profondeur de pénétration s'écrit :

$$\delta(C_1, C_2) = \max \{0, r_1 + r_2 - \|\mathbf{c}_1 - \mathbf{c}_2\|\} \quad (4.30)$$

Appliquée au problème courant, l'équation 4.30 permet de réécrire les termes $f_{ijkl}^{\text{pen}}(\mathbf{v})$:

$$f_{ijkl}^{\text{pen}}(\mathbf{v}) = (\max \{0, r_{ij} + r_{kl} - \|\mathbf{c}_{ij}(\mathbf{v}_i) - \mathbf{c}_{kl}(\mathbf{v}_k)\|\})^2 \quad (4.31)$$

L'expression des contraintes d'appartenance dépend de la géométrie du contenant. Son expression est détaillée ci-après. Pour trouver l'optimum local le plus proche, la méthode quasi-Newton *BFGS* est utilisée. Pour cela, le gradient de la fonction F est évalué :

$$\nabla F(\mathbf{v}) = \omega_{\text{pen}} \nabla F_{\text{pen}}(\mathbf{v}) + \omega_{\text{pro}} \nabla F_{\text{pro}}(\mathbf{v}) \quad (4.32)$$

Les composantes des vecteurs gradients ∇F_{pen} et ∇F_{pro} s'écrivent :

$$\nabla_i F_{\text{pen}}(\mathbf{v}) = \frac{\partial F_{\text{pen}}(\mathbf{v})}{\partial \mathbf{v}_i} = \sum_{j=1}^{n_i} \sum_{k=\{1,\dots,m\} \setminus i} \sum_{l=1}^{n_k} \frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial \mathbf{v}_i} \quad (4.33a)$$

$$\nabla_i F_{\text{pro}}(\mathbf{v}) = \frac{\partial F_{\text{pro}}(\mathbf{v})}{\partial \mathbf{v}_i} = \sum_{j=1}^{n_i} \frac{\partial f_{ij}^{\text{pro}}(\mathbf{v}_i)}{\partial \mathbf{v}_i} \quad (4.33b)$$

Le vecteur gradient de la fonction f_{ijkl}^{pen} nécessaire au calcul de l'équation 4.33a s'écrit :

$$\frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial \mathbf{v}_i} = -2\delta(S_{ij}(\mathbf{v}_i), S_{kl}(\mathbf{v}_k)) \cdot \frac{\partial \mathbf{c}_{ij}(\mathbf{v}_i)^t}{\partial \mathbf{v}_i} \cdot \frac{\mathbf{c}_{ij}(\mathbf{v}_i) - \mathbf{c}_{kl}(\mathbf{v}_k)}{\|\mathbf{c}_{ij}(\mathbf{v}_i) - \mathbf{c}_{kl}(\mathbf{v}_k)\|} \quad (4.34)$$

Les paragraphes suivants présentent les détails spécifiques à chaque dimension du problème, à savoir les dimensions 2 et 3. Le cas 3D fait l'objet d'une attention particulière lorsque le contenant est de forme quelconque.

4.3.3.1 Algorithme de séparation pour les assemblages de cercles

En 2D, le vecteur des inconnues de chaque composant C_i , se décompose de la manière suivante :

$$\mathbf{v}_i = (\mathbf{x}_i, \theta_i)^t = (x_i, y_i, \theta_i)^t \quad (4.35)$$

Le vecteur des inconnues \mathbf{v} a pour dimension \mathbb{R}^{3m} , où m est le nombre de composants. Les coordonnées du centre du cercle S_{ij} sont données par

$$\begin{aligned} \begin{pmatrix} x_{ij} \\ y_{ij} \end{pmatrix} &= \mathbf{c}_{ij}(\mathbf{v}_i) \\ &= \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \begin{pmatrix} p_{ij,x} \\ p_{ij,y} \end{pmatrix} \end{aligned} \quad (4.36)$$

La distance entre les coordonnées des centres \mathbf{c}_{ij} et \mathbf{c}_{kl} des cercles S_{ij} et S_{kl} qui est utilisée pour le calcul des profondeurs de pénétration, s'écrit ici :

$$\|\mathbf{c}_{ij}(\mathbf{v}_i) - \mathbf{c}_{kl}(\mathbf{v}_k)\| = \sqrt{(x_{ij} - x_{kl})^2 + (y_{ij} - y_{kl})^2} \quad (4.37)$$

La fonction f_{ijkl}^{pen} s'écrit donc de la manière suivante pour le problème 2D :

$$f_{ijkl}^{\text{pen}}(\mathbf{v}) = \left(\max \left\{ 0, r_{ij} + r_{kl} - \sqrt{(x_{ij} - x_{kl})^2 + (y_{ij} - y_{kl})^2} \right\} \right)^2 \quad (4.38)$$

Le vecteur gradient de la fonction f_{ijkl}^{pen} associé au composant C_i s'écrit :

$$\frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial \mathbf{v}_i} = \left(\frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial x_i}, \frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial y_i}, \frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial \theta_i} \right)^t \quad (4.39)$$

La matrice jacobienne de la fonction de placement \mathbf{c}_{ij} , nécessaire au calcul du gradient 4.47, se calcule de la manière suivante :

$$\begin{aligned} \frac{\partial \mathbf{c}_{ij}(\mathbf{v}_i)}{\partial \mathbf{v}_i} &= \begin{bmatrix} \frac{\partial x_{ij}}{\partial x_i} & \frac{\partial x_{ij}}{\partial y_i} & \frac{\partial x_{ij}}{\partial \theta_i} \\ \frac{\partial y_{ij}}{\partial x_i} & \frac{\partial y_{ij}}{\partial y_i} & \frac{\partial y_{ij}}{\partial \theta_i} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -\sin(\theta_i) p_{ij,x} - \cos(\theta_i) p_{ij,y} \\ 0 & 1 & \cos(\theta_i) p_{ij,x} - \sin(\theta_i) p_{ij,y} \end{bmatrix} \end{aligned} \quad (4.40)$$

Les composants peuvent être contraints à se déplacer selon une droite ou encore un cercle. L'optimisation locale doit pouvoir prendre en compte cet aspect là. La contrainte de déplacement des composants entraîne une réduction du nombre de degrés de liberté des composants. En 2D, chaque composant a trois degrés de libertés (x_i, y_i, θ_i) . Imposer un déplacement le long d'une droite conduit un composant C_i à n'avoir plus que deux degrés de liberté : (t_i, θ_i) .

$$\begin{aligned} \frac{\partial \mathbf{c}_{ij}(\mathbf{v}_i)}{\partial \mathbf{v}_i} &= \begin{bmatrix} \frac{\partial x_{ij}}{\partial t_i} & \frac{\partial x_{ij}}{\partial \theta_i} \\ \frac{\partial y_{ij}}{\partial t_i} & \frac{\partial y_{ij}}{\partial \theta_i} \end{bmatrix} \\ &= \begin{bmatrix} e_{ix} & -\sin(\theta_i) p_{ij,x} - \cos(\theta_i) p_{ij,y} \\ e_{iy} & \cos(\theta_i) p_{ij,x} - \sin(\theta_i) p_{ij,y} \end{bmatrix} \end{aligned} \quad (4.41)$$

Dans le cas où la droite est horizontale, alors $e_{ix} = 1$, $e_{iy} = 0$. Si la droite est verticale, $e_{ix} = 0$ et $e_{iy} = 1$. Si le déplacement du point de référence \mathbf{r}_i du composant C_i est donné sous une forme paramétrée

$$\begin{cases} x_i(t) = f_x(t) \\ y_i(t) = f_y(t) \end{cases} \quad (4.42)$$

alors,

$$\begin{aligned} \frac{\partial \mathbf{c}_{ij}(\mathbf{v}_i)}{\partial \mathbf{v}_i} &= \begin{bmatrix} \frac{\partial x_{ij}}{\partial t_i} & \frac{\partial x_{ij}}{\partial \theta_i} \\ \frac{\partial y_{ij}}{\partial t_i} & \frac{\partial y_{ij}}{\partial \theta_i} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial f_x(t)}{\partial t} & -\sin(\theta_i) p_{ij,x} - \cos(\theta_i) p_{ij,y} \\ \frac{\partial f_y(t)}{\partial t} & \cos(\theta_i) p_{ij,x} - \sin(\theta_i) p_{ij,y} \end{bmatrix} \end{aligned} \quad (4.43)$$

Les contraintes d'appartenance sont satisfaites si l'intégralité des cercles se trouvent à l'intérieur du contenant. Le non-respect de ces contraintes est évalué à l'aide des profondeurs de pénétration entre le cercle et le contenant. Si le contenant est rectangulaire ou circulaire, les polygones d'appartenance peuvent être facilement évalués. En revanche, si le contenant est quelconque, il faut soit utiliser les méthodes d'*offsets* intérieurs [Wei07], soit approximer chaque cercle par un polygone et calculer le polygone d'appartenance comme présenté en annexe section B.4.2. Un exemple de calcul est donné page 160. Nous choisissons cette seconde méthode, pour laquelle nous disposons de tous les algorithmes nécessaires. Ces calculs doivent être réalisés autant de fois qu'il y a de cercles de rayons différents.

Les figures 4.21 et 4.22 montrent l'influence du degré de liberté en rotation des assemblages de cercles. L'ajout de ce degré de liberté permet à l'algorithme de séparation d'identifier une solution réalisable. La prise en compte de ce degré de liberté dépend bien entendu de la nature du problème. Les poids d'agrégation de la fonction F sont ici identiques : $\omega_{\text{pro}} = 1$ et $\omega_{\text{pen}} = 1$. Les convergences de ces deux exemples sont présentées figure 4.23.

Les différents aspects mathématiques liés à la séparation d'assemblages de cercles viennent d'être discutés. On se propose maintenant de chercher une décomposition en assemblages de cercles qui assure une séparation optimale. L'approximation des composants à l'aide d'assemblages de cercles peut être envisagé de différentes manières. La figure 4.24 présente trois méthodes d'approximations des géométries. Les approches *voxel* et *quadtree* sont des approximations de la géométrie basées sur un découpage régulier de l'espace. L'approche basée sur l'axe médian utilise celui-ci pour placer les différents cercles d'approximation. Une fois que l'axe médian a été calculé, il est échantillonné de manière régulière pour y placer les cercles de rayon maximum. L'échantillonnage peut être calculé en fonction du nombre de cercles désiré. La figure 4.25 présente le problème rencontré lorsqu'une approche *voxel* ou *quadtree* est utilisée avec l'algorithme de séparation. Pour que ce dernier soit efficace, il faut que l'évaluation du gradient de la fonction de séparation propose des directions de séparation entre composants. Ces dernières sont élaborées à partir des gradients des fonctions f_{ij} de l'ensemble des paires de cercles en collision. Chaque paire de cercles en collision cherche uniquement à éliminer son

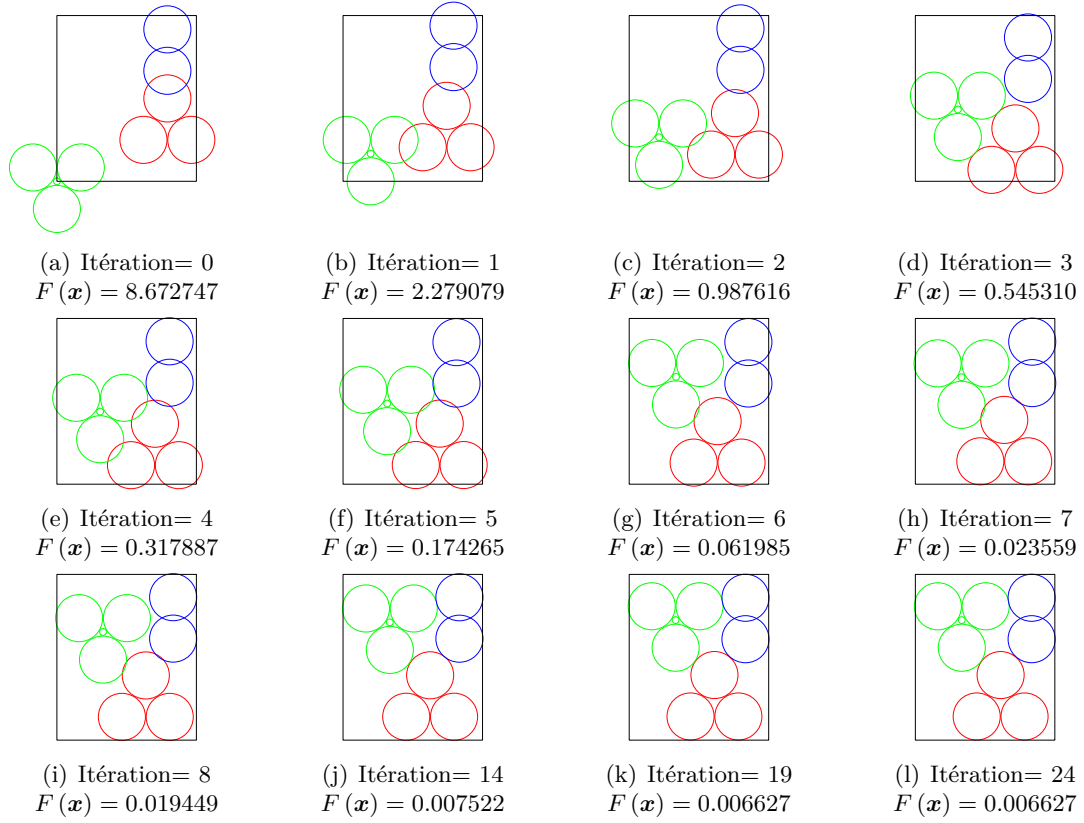


Figure 4.21 – Illustration de l'évolution de la convergence de l'algorithme *BFGS* pour le problème de séparation en translation d'un assemblage de cercles.

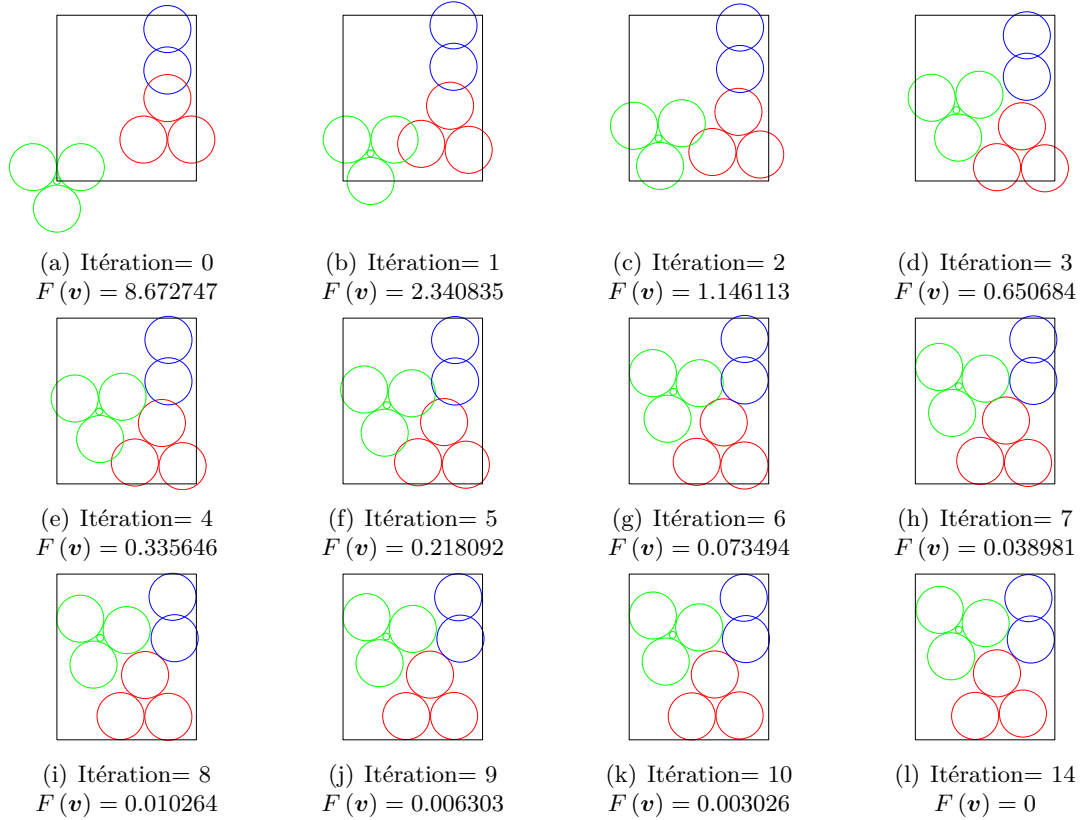


Figure 4.22 – Illustration de l'évolution de la convergence de l'algorithme *BFGS* pour le problème de séparation en translation et rotation d'un assemblage de cercles.

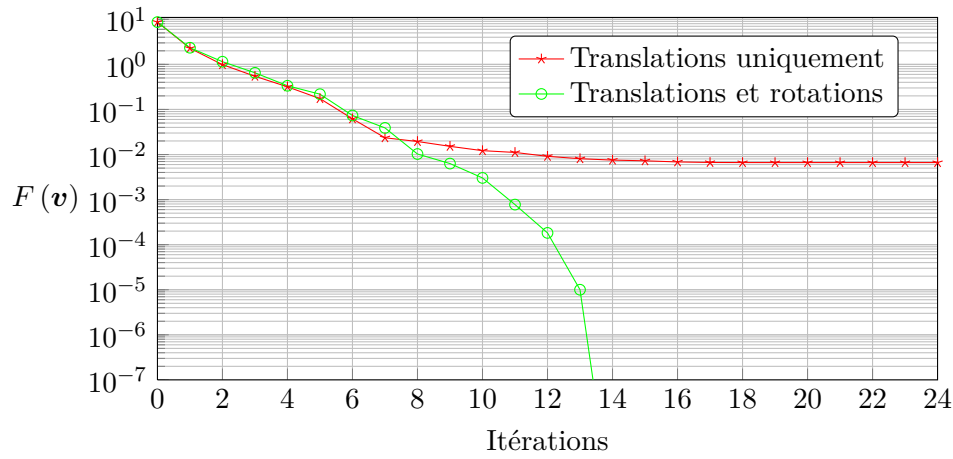


Figure 4.23 – Évolution de la convergence de l'algorithme *BFGS* pour le problème de minimisation de la somme des profondeurs de pénétration d'un assemblage de cercles devant s'inscrire dans un rectangle. Comparaison de l'influence de blocage de certains degrés de liberté.

chevauchement, sans savoir si la contribution qu'elle va apporter au gradient va être globalement positive. Or, sur une décomposition de type *voxel* ou *quadtrees*, les différentes directions de séparation proposées par l'ensemble des paires en collision rentrent facilement en conflit. Les gradients locaux s'annulent, et l'optimiseur s'arrête sur un minimum local. Le fait de changer le rayon des cercles représentant le carré ne change rien au problème rencontré par l'algorithme de séparation. L'utilisation de l'axe médian permet de placer des cercles de rayon maximal dans le composant. De cette manière, chaque direction de séparation est plus facilement orientée vers la frontière des composants.

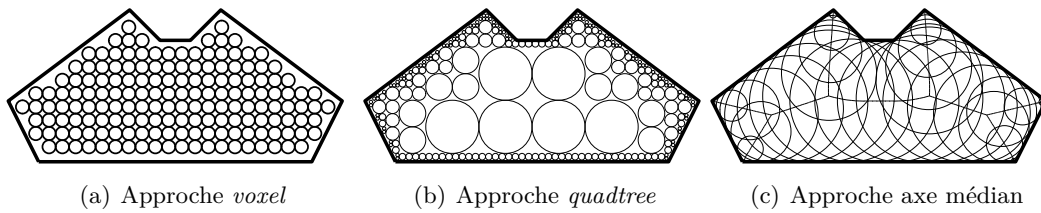


Figure 4.24 – Différents essais de décompositions de polygones en assemblages de cercles.

La figure 4.26 présente une illustration de convergence sur un problème en séparation en rotation avec des composants quelconques. Les composants sont représentés à l'aide d'assemblages de cercles basés sur l'axe médian des composants. La solution initiale a été obtenue en perturbant la solution finale connue.

L'utilisation d'assemblages de cercles pour réaliser la séparation a permis de prendre en compte la libre rotation des composants. Elle permet aussi une généralisation au cas 3D.

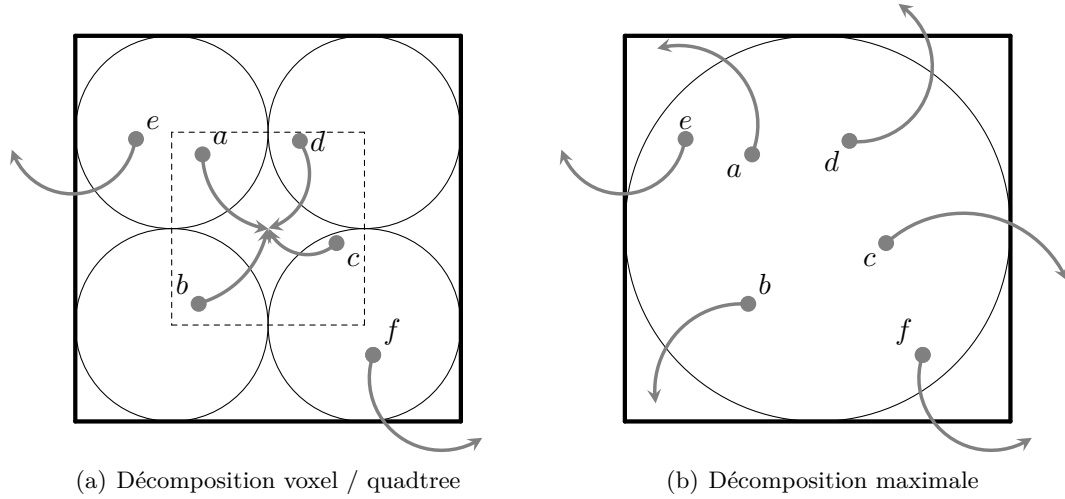


Figure 4.25 – Exemple d’échec pour l’algorithme de séparation avec une décomposition des composants en assemblages de cercles *quadtree*. Sur les deux sous-figures (a) et (b), le composant en forme de carré est respectivement représenté à l’aide de quatre cercles et d’un seul cercle de rayon maximum. Les points gris représentent les positions des centres de cercles chevauchant les différents cercles représentant le carré avant optimisation. Les flèches présentent les positions finales des centres des cercles après séparation. Sous-figure (a), l’ensemble des centres des cercles présents dans le carré en pointillé finira après optimisation au centre de ce carré. L’évaluation des gradients de la fonction de séparation n’a pas fourni de directions de séparation. Les cercles de centres a , b , c et d ne pourront pas être séparés des cercles noirs approximant le carré initial. Seuls les cercles de centres e et f pourront être séparés. Sous-figure (b), ce problème ne se pose pas: les gradients de la fonction de séparation permettent de séparer les cercles se chevauchant.

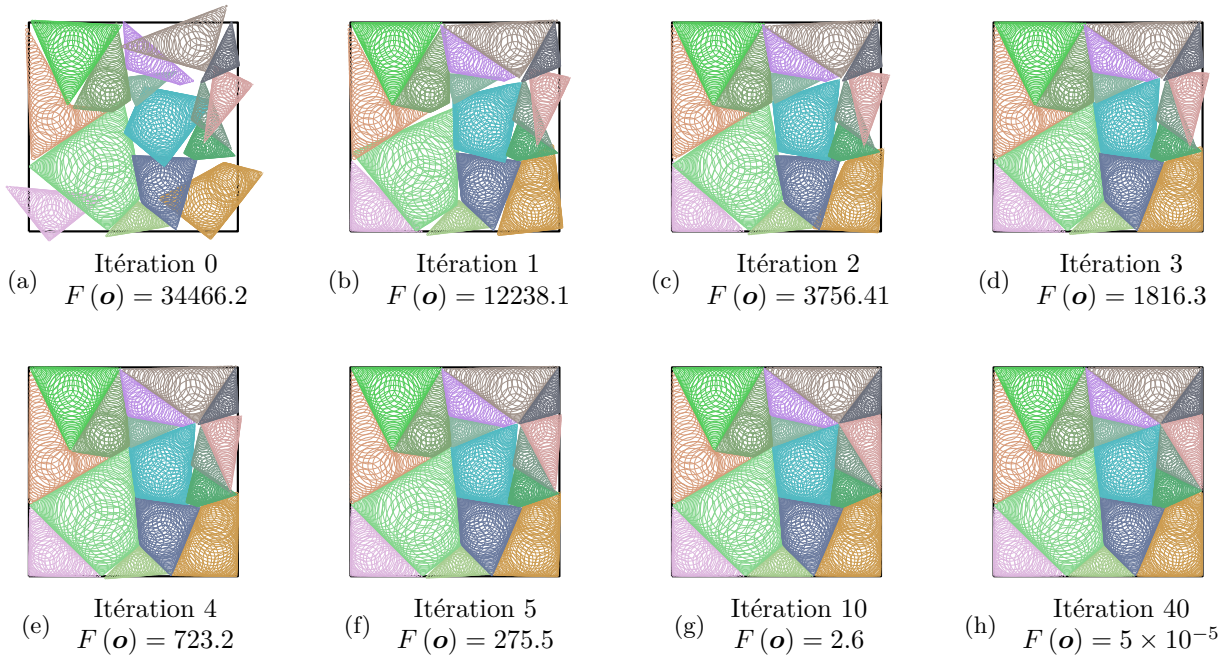


Figure 4.26 – Convergence de l’algorithme d’optimisation *BFGS* pour le problème de séparation des composants sur l’instance *Dighe 1* [DJ96]. La décomposition des composants en assemblages de cercles est basée sur le calcul de l’axe médian des composants. Le seul degré de liberté de chaque composant est une rotation autour de son centre de gravité. Le nombre de variables du problème correspond au nombre de composants, soit 16 variables. Les poids d’agrégation de la fonction F sont identiques : $\omega_{\text{pro}} = 1$ et $\omega_{\text{pen}} = 1$.

4.3.3.2 Algorithme de séparation pour les assemblages de sphères

On présente ici l'algorithme de séparation 3D pour des composants modélisés par des assemblages de sphères devant s'inscrire dans un contenant de géométrie simple : soit un parallélépipède soit une sphère.

Le passage au cas 3D n'entraîne pas de modifications théoriques, les adaptations techniques sont ici présentées. Le repérage des coordonnées est effectué avec les angles d'Euler. Une autre représentation basée sur les quaternions aurait aussi pu être choisie. Pour chaque composant, le vecteur des inconnues en 3D se décompose selon :

$$\mathbf{v}_i = (x_i, y_i, z_i, \phi_i, \theta_i, \psi_i)^t \quad (4.44)$$

Les coordonnées de la sphère S_{ij} sont calculées à partir de la relation suivante :

$$\begin{aligned} \begin{pmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{pmatrix} &= \mathbf{c}_{ij}(\mathbf{v}_i) \\ &= \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + [\mathbf{R}(\phi_i, \theta_i, \psi_i)] \begin{pmatrix} p_{ij,x} \\ p_{ij,y} \\ p_{ij,z} \end{pmatrix} \end{aligned} \quad (4.45)$$

où \mathbf{R} représente la matrice de rotation calculée avec les angles d'Euler. L'écriture de cette matrice est présentée en annexe section B.5. La détection de collision entre deux sphères S_{ij} et S_{kl} se fait à l'aide du calcul de la profondeur de pénétration. La distance entre les centres des sphères S_{ij} et S_{kl} s'écrit :

$$\|\mathbf{c}_{ij}(\mathbf{v}_i) - \mathbf{c}_{kl}(\mathbf{v}_k)\| = \sqrt{(x_{ij} - x_{kl})^2 + (y_{ij} - y_{kl})^2 + (z_{ij} - z_{kl})^2} \quad (4.46)$$

Le vecteur gradient associé à un composant C_i pour le calcul des chevauchements avec le composant C_k s'écrit :

$$\frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial \mathbf{v}_i} = \left(\frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial x_i}, \frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial y_i}, \frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial z_i}, \frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial \phi_i}, \frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial \theta_i}, \frac{\partial f_{ijkl}^{\text{pen}}(\mathbf{v})}{\partial \psi_i} \right)^t \quad (4.47)$$

La matrice jacobienne de la fonction de placement \mathbf{c}_{ij} nécessaire au calcul de l'équation 4.47 s'écrit :

$$\begin{aligned} \frac{\partial \mathbf{c}_{ij}(\mathbf{v}_i)}{\partial \mathbf{v}_i} &= \begin{bmatrix} \frac{\partial x_{ij}}{\partial x_i} & \frac{\partial x_{ij}}{\partial y_i} & \frac{\partial x_{ij}}{\partial z_i} & \frac{\partial x_{ij}}{\partial \phi_i} & \frac{\partial x_{ij}}{\partial \theta_i} & \frac{\partial x_{ij}}{\partial \psi_i} \\ \frac{\partial y_{ij}}{\partial x_i} & \frac{\partial y_{ij}}{\partial y_i} & \frac{\partial y_{ij}}{\partial z_i} & \frac{\partial y_{ij}}{\partial \phi_i} & \frac{\partial y_{ij}}{\partial \theta_i} & \frac{\partial y_{ij}}{\partial \psi_i} \\ \frac{\partial z_{ij}}{\partial x_i} & \frac{\partial z_{ij}}{\partial y_i} & \frac{\partial z_{ij}}{\partial z_i} & \frac{\partial z_{ij}}{\partial \phi_i} & \frac{\partial z_{ij}}{\partial \theta_i} & \frac{\partial z_{ij}}{\partial \psi_i} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & a & b & c \\ 0 & 1 & 0 & d & e & f \\ 0 & 0 & 1 & g & h & i \end{bmatrix} \end{aligned} \quad (4.48)$$

avec

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \frac{\partial \mathbf{c}_{ij}}{\partial \phi_i} = \frac{\partial \mathbf{R}}{\partial \phi_i} \cdot \begin{pmatrix} p_{ij,x} \\ p_{ij,y} \\ p_{ij,z} \end{pmatrix} \quad \begin{pmatrix} d \\ e \\ f \end{pmatrix} = \frac{\partial \mathbf{c}_{ij}}{\partial \theta_i} = \frac{\partial \mathbf{R}}{\partial \theta_i} \cdot \begin{pmatrix} p_{ij,x} \\ p_{ij,y} \\ p_{ij,z} \end{pmatrix} \quad \begin{pmatrix} g \\ h \\ i \end{pmatrix} = \frac{\partial \mathbf{c}_{ij}}{\partial \psi_i} = \frac{\partial \mathbf{R}}{\partial \psi_i} \cdot \begin{pmatrix} p_{ij,x} \\ p_{ij,y} \\ p_{ij,z} \end{pmatrix} \quad (4.49)$$

Voici en détails les expressions des différents gradients pour une variable de translation et une variable de rotation :

$$\begin{aligned} \frac{\partial f_{ijkl}^{\text{pen}}}{\partial x_i} &= 2\beta_{\text{pen}}(x_{ij} - x_{kl}) \\ \frac{\partial f_{ijkl}^{\text{pen}}}{\partial \phi_i} &= 2\beta_{\text{pen}} \left(\frac{\partial \mathbf{R}(\phi_i, \theta_i, \psi_i)}{\partial \phi_i} \cdot \begin{pmatrix} p_{ij,x} \\ p_{ij,y} \\ p_{ij,z} \end{pmatrix} \right)^t \cdot (\mathbf{c}_{ij}(\mathbf{v}_i) - \mathbf{c}_{kl}(\mathbf{v}_k)) \end{aligned} \quad (4.50)$$

avec

$$\beta_{\text{pen}} = -\delta(S_{ij}(\mathbf{v}_i), S_{kl}(\mathbf{v}_k)) / \|\mathbf{c}_{ij}(\mathbf{v}_i) - \mathbf{c}_{kl}(\mathbf{v}_k)\| \quad (4.51)$$

Les dérivées des matrices de rotation sont présentées en annexe section B.5. Les contraintes d'appartenance g_i sont calculées à partir des profondeurs de pénétration des sphères avec le complémentaire du contenant \bar{C} , comme indiqué équation 4.29b. Ces contraintes peuvent être facilement évaluées lorsque le contenant est géométriquement simple, comme un parallélépipède ou une sphère. Les contenants de géométrie complexe font l'objet de la prochaine section.

La figure 4.27 présente un exemple de séparation de 1000 sphères. Avant de montrer un exemple de séparation d'assemblages de sphères, il faut convertir les composants en assemblages de sphères. Comme en 2D, il existe de nombreuses techniques pour approximer un objet 3D à l'aide de sphères. Les techniques les plus simples consistent à utiliser des représentations *voxels* ou *octrees*. Toutefois, pour éviter les problèmes de convergence prématurée rencontrée en 2D et pour limiter le nombre de sphères utilisées, les décompositions seront basées sur des axes médians. Pour cela, on se base sur les travaux de Hubbard [Hub96] et Bradshaw *et al.* [BO02, BO04]. Initialement conçue pour la détection de collisions en temps réel, la technique développée par Hubbard utilise une approximation de l'axe médian des composants pour positionner les sphères et construire une représentation hiérarchique du composant. Bradshaw *et al.* ont proposé plusieurs améliorations qui permettent d'obtenir des approximations encore plus fidèles. La figure 4.28 présente la transformation d'un modèle 3D en assemblages hiérarchiques de sphères, basée sur une approximation de l'axe médian du modèle². Chaque niveau n contient 8^n sphères et représente un raffinement du niveau supérieur. Le niveau 0 contient la sphère englobante du modèle. Le dernier niveau, qui approxime au mieux chaque composant, comporte $8^3 = 512$ sphères et sera utilisé pour les tests à venir. La figure 4.29 présente un exemple de séparation en translation de neuf assemblages de sphères identiques dans un contenant parallélépipédique. Les translations des composants permettent de minimiser au mieux le non-respect de la contrainte de non-chevauchement.

Enfin, les décompositions hiérarchiques permettent de mettre en place des algorithmes de détection de collisions récursifs pour diminuer le nombre de tests à effectuer. La figure 4.30 présente en 2D les différents tests d'intersection effectués pour déterminer la collision ou non de deux composants. Pour faciliter la compréhension de fonctionnement de ces algorithmes, la décomposition hiérarchique est ici basée sur un *quadtrees*.

4.3.3.3 Algorithme de séparation 3D pour composants et contenants quelconques

Dans la majorité des problèmes, le contenant n'est pas parallélépipédique mais quelconque, comme pour les composants. Nous proposons ici une technique pour prendre en compte les contenants de géométries quelconques dans l'algorithme de séparation.

Comme précédemment, les composants sont représentés à l'aide d'assemblages de sphères calculés à partir des approximations des axes médians des composants. La détection des chevauchements entre les sphères des composants se fait à l'aide des calculs de profondeurs de pénétration.

La gestion des contraintes d'appartenance au contenant ne peut plus se faire à l'aide de bornes, ou des volumes d'appartenance qui ne sont plus de géométries simples comme auparavant. Les volumes d'appartenance dans le cas quelconque sont obtenus à l'aide de calculs d'*offsets* intérieurs très coûteux [Wei07]. On se propose de calculer les profondeurs de pénétration approchées entre les sphères et la surface du contenant pour évaluer le non-respect des contraintes d'appartenance.

Le contenant de géométrie quelconque est représenté sous la forme d'un polyèdre. Pour faciliter les calculs, chaque face polygonale est triangulée. L'ensemble des triangles représente la frontière du modèle 3D. Le contenant sera représenté informatiquement par la liste des coordonnées des sommets des triangles et une table de connectivité des triangles, comme pour les formats de fichiers *stl* ou *obj*.

². Le modèle et sa décomposition hiérarchique sont disponibles en téléchargement à l'adresse <http://isg.cs.tcd.ie/spheretree/>

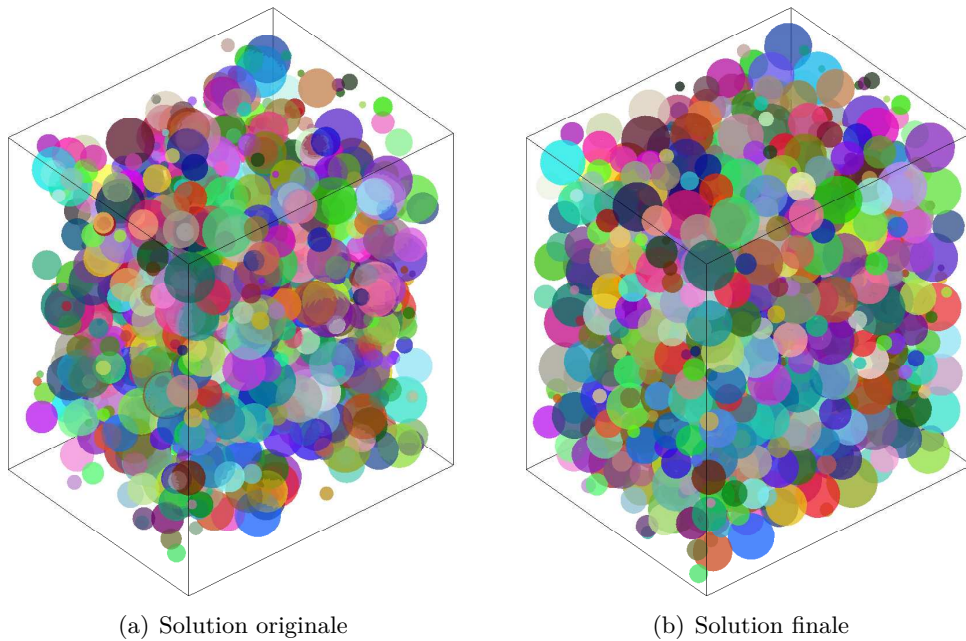


Figure 4.27 – Convergence de l’algorithme d’optimisation *BFGS* pour le problème de séparation en translation de 1000 sphères. Le nombre de variables du problème correspond à trois fois le nombre de sphères, soit 3000 variables. Les poids d’agrégation de la fonction F sont identiques : $\omega_{\text{pro}} = 1$ et $\omega_{\text{pen}} = 1$. La compacité du problème est de 53.6%.

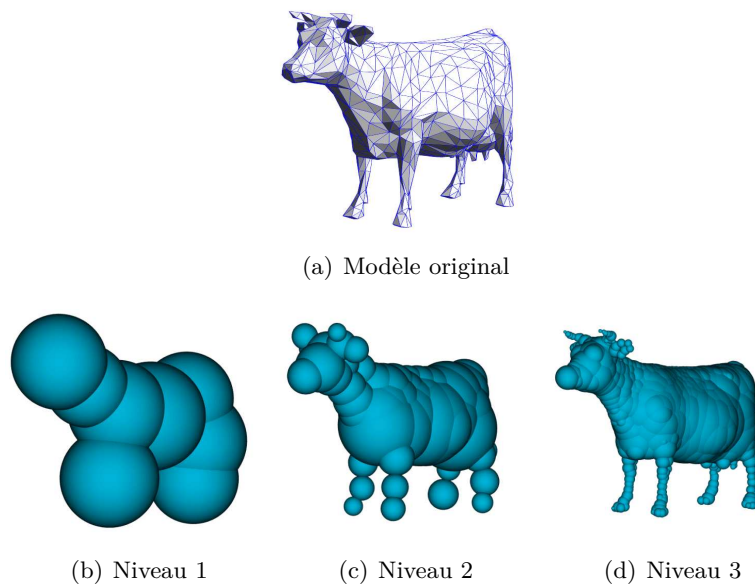


Figure 4.28 – Transformation d’un modèle 3D en assemblages hiérarchiques de sphères basés sur l’axe médian. Image empruntée à Bradshaw *et al.* [BO04].

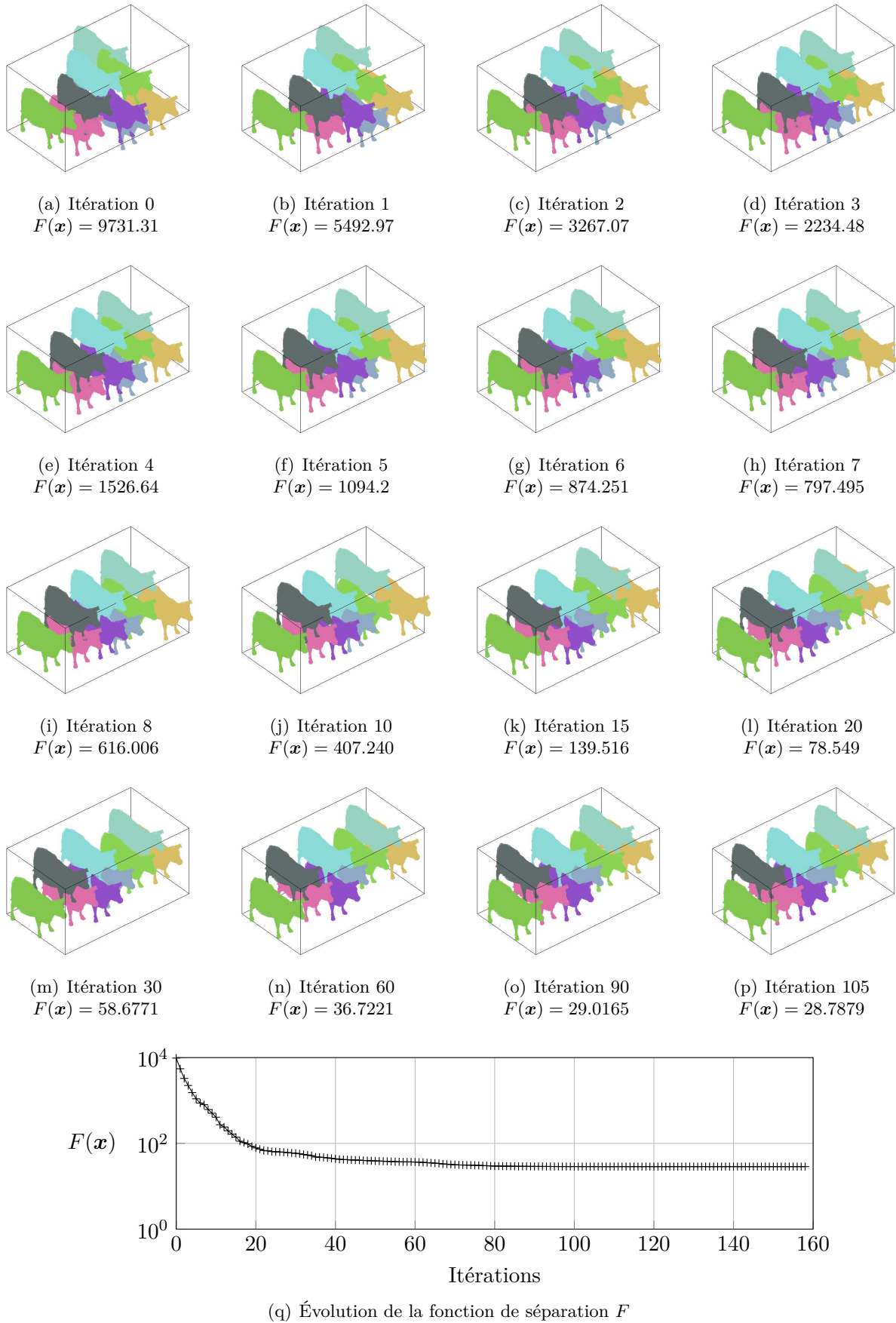
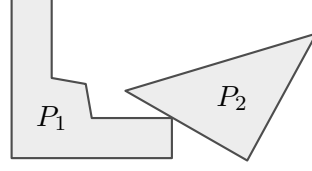
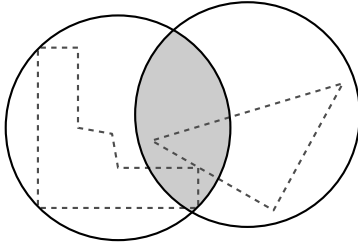
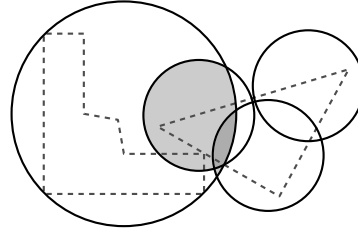


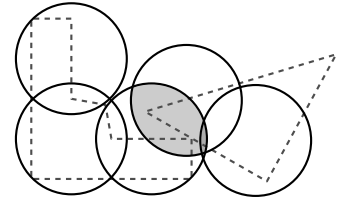
Figure 4.29 – Convergence de l’algorithme d’optimisation *BFGS* pour le problème de séparation en translation de neuf assemblages de sphères identiques. Le nombre de variables du problème correspond à trois fois le nombre d’assemblages de sphères, soit 27 variables. La sous-figure (q) montre l’évolution de la fonction F .


 (a) Polygones originaux P_1 et P_2

 (b)

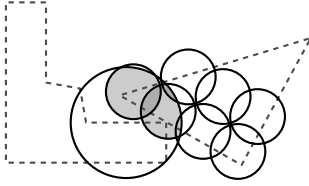
	P_1	P_2
Niveau	0	0


 (c)

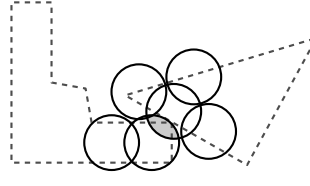
	P_1	P_2
Niveau	0	1


 (d)

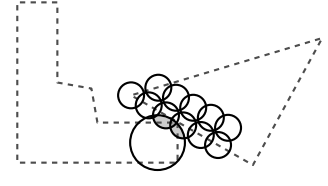
	P_1	P_2
Niveau	1	1


 (e)

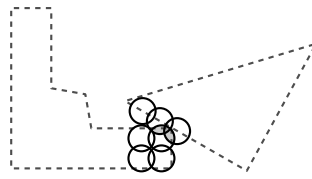
	P_1	P_2
Niveau	1	2


 (f)

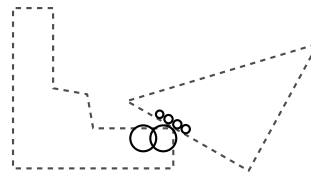
	P_1	P_2
Niveau	2	2


 (g)

	P_1	P_2
Niveau	2	3


 (h)

	P_1	P_2
Niveau	3	3


 (i)

	P_1	P_2
Niveau	3	4

Figure 4.30 – Exemple de détection de collisions à l'aide d'une représentation hiérarchique. Les polygones originaux sont approximés hiérarchiquement à l'aide de cercles à partir d'une décomposition *quadtree*. Chaque carré du *quadtree* a été remplacé par son cercle circonscrit, sauf sur le dernier niveau où le carré est remplacé par son cercle inscrit.

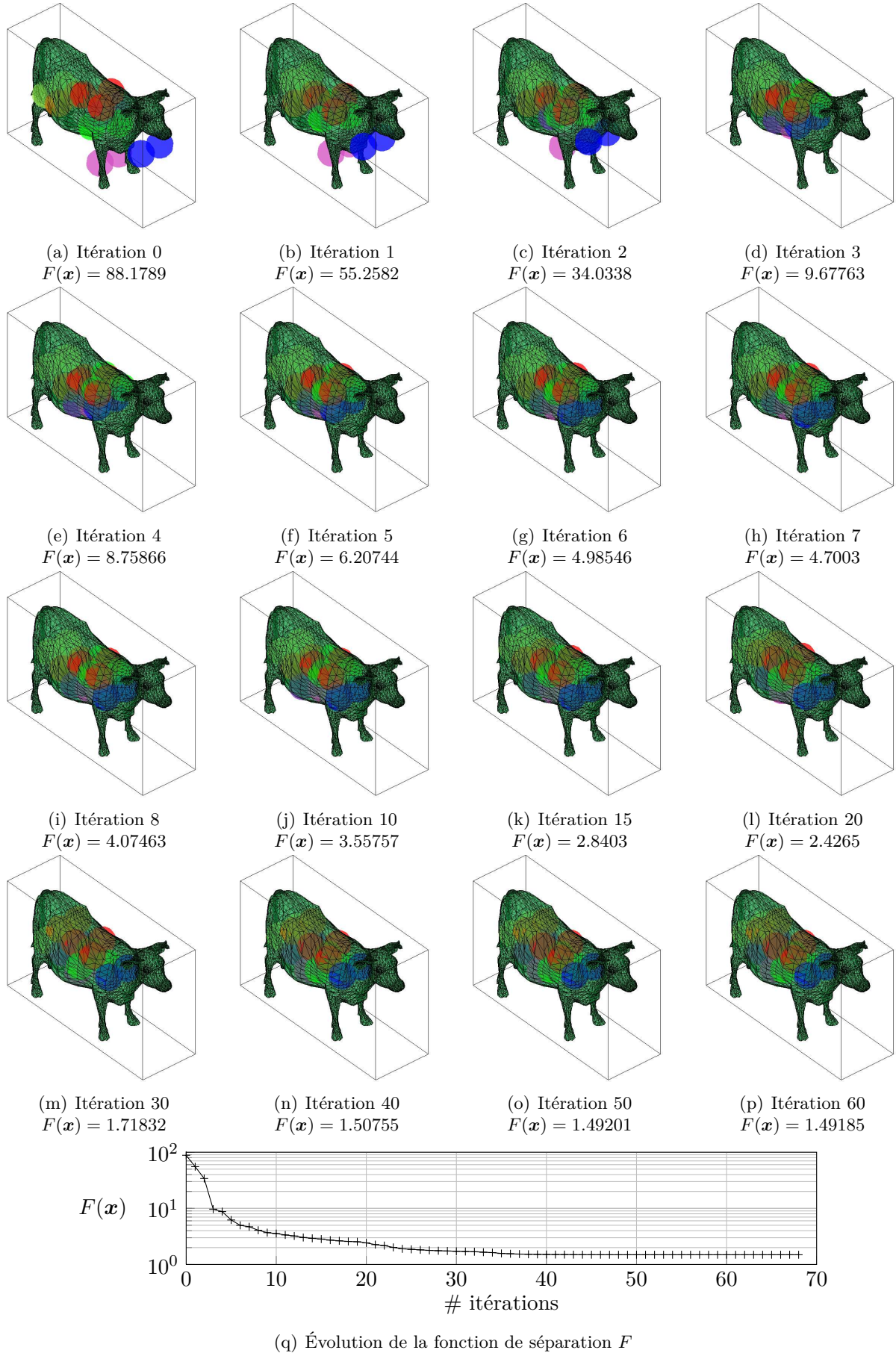


Figure 4.31 – Convergence de l’algorithme d’optimisation *BFGS* pour le problème de séparation en translation de sept assemblages de sphères dans un contenant de géométrie complexe (21 variables). La sous-figure (q) montre l’évolution de la fonction F .

La détection de l'appartenance des sphères au contenant repose sur deux tests géométriques :

- le premier est un test d'appartenance d'un point à une sphère, qui se vérifie à l'aide d'un calcul de distance,
- le second est un test d'appartenance d'un point \mathbf{p} à un polyèdre. Comme en 2D, ce test est basé sur le nombre d'intersections de triangles d'un rayon semi-infini partant du point \mathbf{p} à tester. Une étape initiale consiste à tester si le point \mathbf{p} est sur le contenant. Cela revient à vérifier si le point appartient à un ensemble de triangles 3D. Si le point n'appartient pas à la surface du contenant, alors un test basé sur un rayon semi-infini partant du point \mathbf{p} est utilisé. La direction de ce rayon est fixée initialement comme une direction principale de la scène (axe x , y ou z). Cette direction principale permet d'éliminer rapidement toute une série de triangles, à l'aide de tests de boîtes englobantes. Si ce rayon intersecte un nombre impair de triangles, alors le point \mathbf{p} est à l'intérieur du contenant. Dans le cas contraire, il se trouve à l'extérieur. Si ce rayon intersecte un triangle sur un de ses sommets ou arrêtes, alors la direction du rayon est changée aléatoirement de manière à éviter toute ambiguïté avec les sommets ou encore avec les segments du contenant.

La détection de l'appartenance des sphères au contenant se fait en deux temps : la première étape consiste à vérifier que les centres des sphères sont situés dans le contenant. La deuxième étape consiste à vérifier que l'intégralité de chaque sphère réside dans le contenant.

On présente ici les calculs à réaliser pour évaluer la fonction de séparation F et s'assurer qu'une sphère se trouve bien à l'intérieur du contenant. Dans un premier temps, un test d'appartenance du centre de la sphère au contenant est réalisé :

- si le centre de la sphère considérée est sur ou à l'extérieur du contenant alors, on minimise la distance au carré entre ce centre et le centre du contenant. Cette pénalité est pondérée avec le volume de la sphère.
- si le centre est à l'intérieur du modèle, on doit s'assurer que la sphère appartient au contenant. Pour cela, on utilise une méthode approchée. Un ensemble de points répartis de manière uniforme est généré sur le contenant. Ces points sont utilisés pour s'assurer que l'ensemble des sphères sont entièrement contenus dans le contenant. Si une partie de ces points est contenue dans la sphère, alors la sphère n'est pas complètement contenue dans le contenant. On pénalise la fonction de séparation de la profondeur de pénétration au carré entre ces points et la sphère. Si \mathbf{p} est un point contenu dans la sphère de centre \mathbf{c}_{ij} et de rayon r_{ij} , la profondeur de pénétration s'écrit

$$\delta = \max \{0, r_{ij} - |\mathbf{c}_{ij} - \mathbf{p}|\} \quad (4.52)$$

De la qualité de répartition de ces points dépend la qualité de prise en compte des composants à l'intérieur du contenant.

La minimisation de la fonction de séparation est toujours réalisée avec l'algorithme *BFGS*. La figure 4.31 présente un exemple de séparation en translation de sept assemblages de sphères dans un contenant de géométrie complexe. Le problème compte donc 21 variables.

4.3.4 Récapitulatif des différentes variantes de l'algorithme de séparation

Les différentes variantes de l'algorithme de séparation sont résumées table 4.5.

	Composants	Contenant	Type de séparation
	Rectangles	Rectangle	Min. du produit aire intersection profondeur de pénétration
2D	Cercles	Rectangle / Cercle	Min. pénétration
	Polygones à orient. discrètes	Polygone	Min. pénétration avec NFP/IFP
	Polygones à orient. continues	Polygone	Min. pénétration avec ass. cercles
	Parallélépipèdes	Parallélépipède	Min. du produit volume intersec- tion profondeur de pénétration
3D	Sphères	Parallélépipède / Sphère	Min. pénétration
	Polyèdres	Parallélépipèdes	Min. pénétration avec ass. sphères
	Polyèdres	Polyèdre	Min. pénétration avec ass. sphères et nuage de points

Table 4.5 – Résumé des différentes variantes de l'algorithme de séparation. Les abréviations utilisées dans cette table sont les suivantes : Min. : Minimisation. Ass. : Assemblages. Orient. : Orientations. L'expression Min. pénétration est un raccourci pour minimisation des profondeurs de pénétration.

4.4 Conclusion

Hybridation d'un algorithme génétique et d'un algorithme de séparation, la méthode proposée a été développée dans l'optique de fournir un cadre de travail ouvert pour la résolution des problèmes de placement. L'algorithme génétique a pour objectif de réaliser l'optimisation globale en proposant des solutions prometteuses, à l'aide des opérateurs de sélection, croisement, mutation et échange. L'algorithme de séparation modifie localement ces solutions pour obtenir des solutions réalisables. Basé sur la minimisation d'une fonction continue, cet algorithme s'adapte à la géométrie des composants ainsi qu'à leurs degrés de libertés. Dans le cas de composants rectangulaires ou parallélépipédiques, nous avons montré que la minimisation directe de l'aire ou du volume de chevauchement ne permettait pas d'obtenir une séparation optimale. Il vaut mieux minimiser le produit des profondeurs de pénétration avec les volumes de chevauchement. Si le problème est bidimensionnel et l'orientation des composants est discrète, l'algorithme de séparation est basé sur l'utilisation des polygones de non-recouvrement et appartenance et cherche à minimiser les profondeurs de pénétration entre composants. Dès lors que l'on souhaite prendre en compte la libre rotation des composants, ou bien que le problème est tridimensionnel, les composants sont représentés à l'aide d'assemblages de cercles / sphères, pour lesquels on cherche à minimiser les profondeurs de pénétration entre chaque paire de sphères. Pour obtenir des algorithmes de séparation efficaces, nous avons montré qu'il valait mieux construire les assemblages de sphères à partir des axes médians des composants. Nous avons proposé un algorithme de séparation 3D pour des contenants de géométries complexes. Cet algorithme devra par la suite être intégré à l'optimiseur global pour obtenir un solveur complet pour les problèmes 3D.

Enfin, la méthode permet un éventail de possibilités qui n'ont pu toutes être exploitées, comme :

- l'intégration de contraintes propres à l'agencement (alignement de composants, mise à distance, placement relatif, ...)
- la possibilité de prendre en compte des composants articulés ;
- la possibilité de prendre en compte des composants à géométrie variable, comme un rectangle dont l'aire serait définie mais pas les dimensions.

L'analyse du comportement de la méthode de placement proposée sur différents exemples fait l'objet du chapitre suivant.

Chapitre 5

Résolution de problèmes de placement bidimensionnels

Dans ce chapitre, la méthode de placement proposée est testée sur trois exemples de placement bidimensionnels. Le premier est une instance de la littérature d'un problème de découpe de formes irrégulières. Le second et le troisième sont des problèmes d'agencement que nous avons créés, sur lesquels est testée la méthode proposée.

Sommaire

5.1	Résultats obtenus sur un problème de découpe de formes irrégulières	102
5.2	Résultats obtenus pour un 1^{er} problème d'agencement	103
5.2.1	Données du problème	103
5.2.2	Premières optimisations	105
5.2.3	Solutions du front de Pareto	111
5.2.4	Rôle de l'algorithme de séparation	112
5.2.5	Rôle des opérateurs génétiques	115
5.2.6	Réglage des opérateurs génétiques	116
5.2.7	Choix de l'algorithme d'optimisation globale	123
5.2.8	Rôle de la population initiale	126
5.2.9	Discussion	128
5.3	Résultats obtenus pour un 2^{ème} problème d'agencement	128
5.3.1	Données du problème	128
5.3.2	Premières optimisations	129
5.3.3	Solutions du front de Pareto	131
5.3.4	Réglages des opérateurs génétiques	131
5.3.5	Relaxation des contraintes de placement	133
5.4	Conclusion	135

5.1 Résultats obtenus sur un problème de découpe de formes irrégulières

Dans ce premier exemple, on compare les résultats obtenus avec notre méthode et les solutions de référence pour un problème de découpe de formes irrégulières. La méthode proposée n'est pas dédiée à ce genre de problèmes. Toutefois, il est intéressant de voir comment elle se comporte.

L'objectif d'un tel problème est de découper un ensemble de composants dans un contenant rectangulaire de largeur fixée et de longueur minimale. Une des techniques de résolution de ces problèmes consiste à générer une solution réalisable et à essayer de l'améliorer de manière itérative. Classiquement, la longueur du contenant de la première solution réalisable est utilisée comme borne supérieure du problème. Ensuite, cette borne est réduite, la solution proposée ne respecte plus les contraintes d'appartenance du problème. L'objectif est alors de résoudre le problème de décision suivant : *est-il possible de placer l'ensemble des composants tel qu'il satisfasse les contraintes de placement ?* Si oui, la borne est encore réduite et le processus est recommencé, sinon la borne est agrandie pour faciliter la résolution du problème. Le calcul s'arrête dès lors qu'un temps limite est dépassé, ou que l'algorithme de résolution ne propose plus d'amélioration suffisante.

On compare nos résultats sur l'instance ALBANO, tirée de l'industrie textile [AS80]¹. Chacun des 24 composants peut avoir quatre orientations discrètes réparties tous les 90°. Le problème comporte donc 48 variables continues et 24 variables discrètes.

Les résultats de notre méthode sont comparés avec trois autres méthodes de la littérature. Les deux premières sont les méthodes *GLSHA* (*Greedy Local Search Hybrid Algorithm*) et *SAHA* (*Simulated Annealing Hybrid Algorithm*) proposées par Gomes *et al.* [GO06]. Ce sont deux méthodes hybrides similaires basées sur l'utilisation de la programmation linéaire pour compacter et séparer les solutions. *GLSHA* utilise une heuristique améliorée de type *Bottom-Left* pour réaliser le placement global des composants, alors que *SAHA* utilise le recuit simulé [KGV83]. La troisième correspond à la méthode proposée par Bennell *et al.* [BS07]. C'est une méthode de placement légal couplée à un algorithme d'optimisation déterministe de recherche par faisceaux (*Beam search*). Ce dernier présente l'avantage de pouvoir contrôler les temps d'exécution à l'aide de deux paramètres : la largeur de filtre (α) et la largeur de faisceaux (β).

La figure 5.1 présente les résultats graphiques des différentes méthodes appliquées à l'instance ALBANO. Les sous-figures (a), (b), (c) présentent les résultats de la littérature. La sous-figure (d) présente la solution obtenue avec notre méthode. La table 5.1 résume numériquement tous les résultats avec la compacité des solutions trouvées ainsi que la longueur de stock nécessaire à la découpe des composants. Les temps de calcul des différents exemples sont respectivement 209s, 2257s, 5460s et 1317s sur des ordinateurs de puissance équivalente.

Solution	Méthode	Réf.	Compacité	Longueur	Temps de calcul
<i>a</i>	<i>GLSHA</i>	[GO06]	86.41%	10074.09	209s
<i>b</i>	<i>SAHA</i>	[GO06]	87.42%	9957.41	2257s
<i>c</i>	<i>Beam Search</i>	[BS07]	87.88%	9906.44	5460s
<i>d</i>	GA + Algo. séparation		86.62%	10050.00	1317s

Table 5.1 – Récapitulatif des meilleurs résultats obtenus pour les différentes méthodes de résolution des problèmes de découpe de formes irrégulières sur l'instance ALBANO. La meilleure solution, mais plus longue à calculer, est la *c* avec une compacité de 87.88%

Notre solution est à 1.4% de la solution de référence, ce qui est correct pour une méthode non dédiée à ce type de problème. De plus, l'initialisation de notre solution est aléatoire. La qualité de nos résultats aurait pu être améliorée en utilisant une heuristique de type *Bottom-Left* pour initialiser les solutions.

Avec notre méthode et pour un problème de découpe de formes irrégulières, l'ensemble de la population finit par converger vers un ensemble de solutions très proches les unes des autres. Le problème

1. Les données de cette instance sont disponibles sur le site Internet de l'ESICUP (*EURO Special Interest Group on Cutting and Packing*) <http://www.fe.up.pt/esicup>

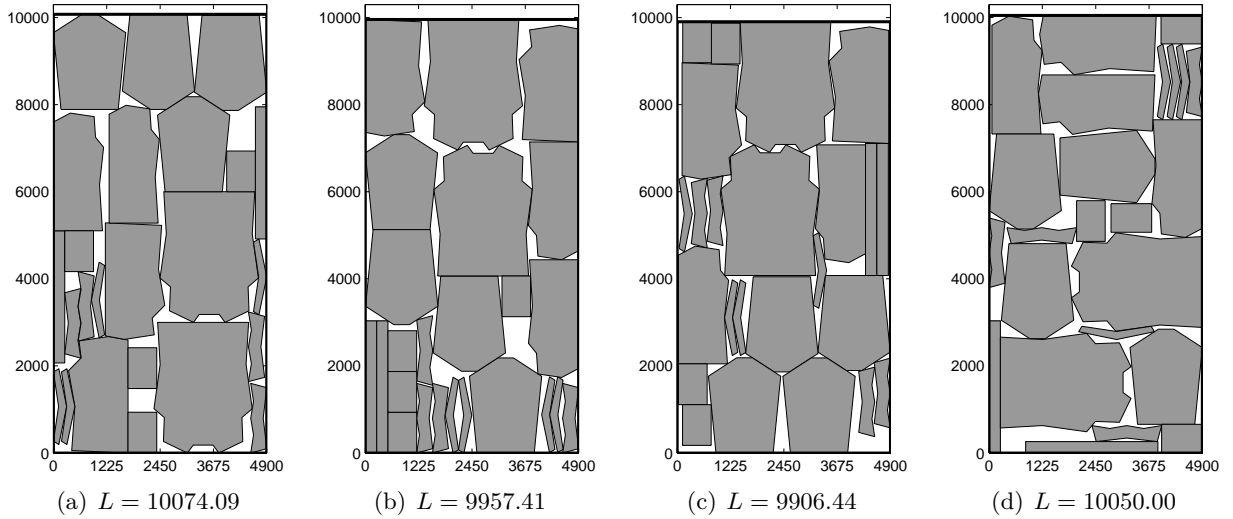


Figure 5.1 – Comparaison des résultats obtenus avec notre méthode et les autres solutions de référence de la littérature. La différence entre la meilleure solution (c) et notre solution (d) est de 1.4% en notre défaveur.

étant mono-objectif, quasiment toutes les solutions présentent un rang différent. Les calculs de distances génotypique et phénotypique dans l'algorithme *Omni-Optimizer* ne permettent donc pas de contribuer à la sélection de solutions diverses. Des études plus poussées devraient être menées pour éviter cette perte de diversité, notamment sur le coefficient de relaxation de dominance. Une relaxation du critère de dominance plus importante permettrait de maintenir dans la génération courante davantage de solutions efficaces. Cela permettrait de faire évoluer en parallèle plusieurs solutions prometteuses et ainsi d'éviter de faire stagner les solutions dans les minima locaux comme ça peut être le cas actuellement. Enfin, l'utilisation d'un algorithme générationnel n'est pas recommandée : en effet, une solution intéressante ou prometteuse ne peut pas être directement réutilisée pour générer de nouvelles solutions. Il faut attendre la génération suivante pour qu'elle puisse être sélectionnée. Il vaudrait mieux utiliser un algorithme à génération continue, dans lequel les solutions nouvellement créées peuvent directement être réutilisées pour la création de nouvelles solutions [DNLA09]. Le surcoût lié aux calculs supplémentaires de rangs des solutions est négligeable face au gain en vitesse de convergence.

La méthode proposée n'est pas spécialisée pour la résolution de ces problèmes, toutefois elle permet de s'approcher des résultats de référence aussi bien en termes de temps de calcul que de qualité de résultat. Quelques adaptations et études permettraient de rendre la méthode plus robuste et efficace pour ce genre de problèmes.

Pour tester notre méthode sur des problèmes plus génériques, deux problèmes d'agencement 2D multi-objectifs ont été créés. Les sections suivantes étudient en détails les résultats obtenus ainsi que le réglage des algorithmes.

5.2 Résultats obtenus pour un 1^{er} problème d'agencement

Soit un problème de chargement de coffre de voiture dans lequel une série de composants doit être placée. L'objectif du problème est d'agencer l'ensemble des composants tel que l'inertie de l'assemblage soit minimale et que les distances entre certains composants soient maximales.

5.2.1 Données du problème

On considère un problème bi-objectif avec 10 composants. La figure 5.2 présente les données géométriques pour cet exemple d'agencement. La sous-figure (a) présente la géométrie du contenant, la sous-figure (b) présente les différents types de composants à positionner. Les coordonnées des points

de l'ensemble des polygones du problème sont listées table 5.2. La table 5.3 donne les caractéristiques des différents composants, ainsi que le nombre de composants pour chacun des types de composants. Les composants de même type ont les mêmes propriétés. La densité des composants est représentée par le niveau de gris des composants : plus un composant est foncé, plus sa densité est grande.

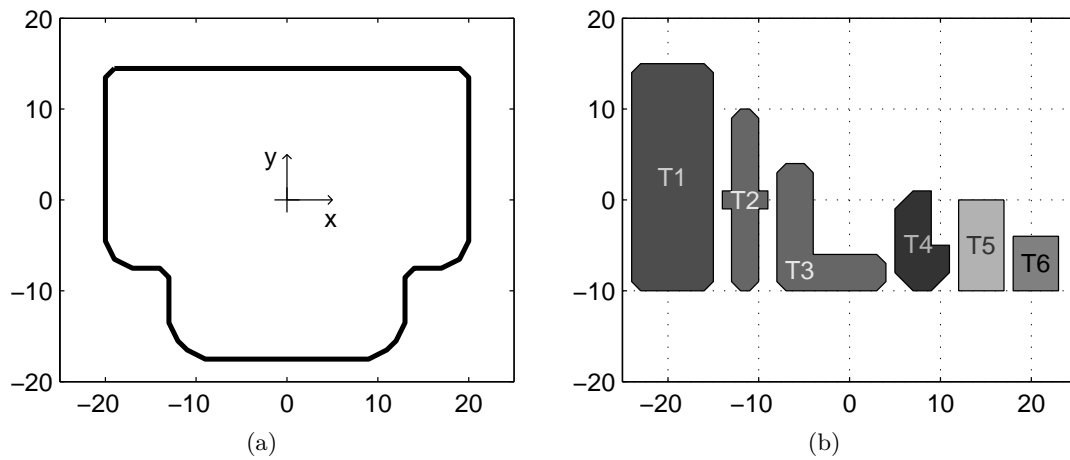


Figure 5.2 – Géométrie du contenant et des différents types de composants à positionner.

Polygones	Coordonnées des points des polygones																			
C	x	1	0	0	1	3	6	7	7	8	9	11	29	31	32	33	33	34	37	39
	y	32	31	13	11	10	10	9	4	2	1	0	0	1	2	4	9	10	10	11
$T1$	x	1	8	9	9	8	1	0	0											
	y	0	0	1	24	25	25	24	1											
$T2$	x	4	3	2	1	1	0	0	1	1	2	3	4	4	5	5	4			
	y	1	0	0	1	9	9	11	11	19	20	20	19	11	11	9	9			
$T3$	x	0	0	1	11	12	12	11	4	4	3	1								
	y	13	1	0	0	1	3	4	4	13	14	14								
$T4$	x	0	0	2	4	4	6	6	4	2										
	y	2	9	11	11	5	5	2	0	0										
$T5$	x	5	0	0	5															
	y	10	10	0	0															
$T6$	x	0	5	5	0															
	y	6	6	0	0															

Table 5.2 – Coordonnées des polygones utilisés pour les problèmes d'agencements.

Type de composants		T1	T2	T3	T4	T5	T6
Nombre de composants	m_i	1	2	1	1	2	3
Aire des composants	A_i	223	62	85.5	48	50	30
Densité des composants	ρ_i	0.7	0.6	0.6	0.8	0.3	0.5

Table 5.3 – Propriétés des composants

Les 10 composants à positionner sont présentés figure 5.3. La position des composants n'est pas contrainte à l'intérieur du contenant. Chaque composant a une orientation discrète comprise dans l'ensemble $\{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$, sauf pour les composants symétriques qui ont seulement deux orientations possibles $\{0^\circ, 90^\circ\}$. Un nombre plus important de rotations aurait pu être pris en compte, avec des orientations quelconques. Toutefois, la géométrie des composants est fortement orientée selon les axes du repère (les axes principaux d'inertie sont confondus dans deux cas sur trois avec les axes du repère).

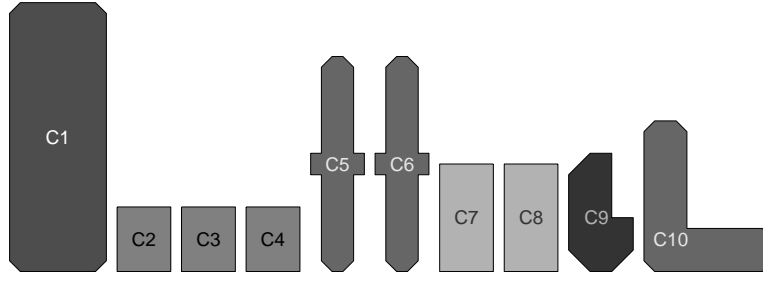


Figure 5.3 – Présentation des différents composants à positionner.

Avec 10 composants, le problème possède 20 variables réelles qui codent la position des composants et 10 variables discrètes représentant l'orientation discrètes des composants. Avec un contenant d'une aire de 1123, la compacité du problème est de 59.7%.

Les contraintes de placement, qui regroupent ici les contraintes de non-chevauchement ainsi que les contraintes d'appartenance, sont exprimées en terme de profondeur de pénétration. Elles sont regroupées dans une seule et même fonction que l'algorithme de séparation doit minimiser. Les bornes de chaque variable continue sont calculées à partir des boîtes englobantes des polygones d'appartenance.

Les objectifs de ce problème consistent à minimiser l'inertie I_{yy} autour de l'axe de symétrie du contenant, et maximiser les distances euclidiennes minimales respectivement entre les deux composants de type $T2$, ainsi que ceux de type $T5$. Dans cet exemple, les deux composants de type $T2$ sont notés C_5 et C_6 , et ceux de type $T5$ sont notés C_7 et C_8 (Voir figure 5.3). Les objectifs du problème s'écrivent mathématiquement :

$$\mathcal{P} \begin{cases} \min f_1(\mathbf{v}) = \sum_{i=1}^m \rho_i I_{i/yy} \\ \max f_2(\mathbf{v}) = d_{\min}(C_5, C_6) + d_{\min}(C_7, C_8) \\ \text{s.c. les contraintes de placement soient satisfaites} \end{cases} \quad (5.1)$$

où m désigne le nombre de composants du problème, $I_{i/yy}$ caractérise l'inertie de surface du composant i autour de l'axe y .

$$I_{i/yy} = \iint_{(x,y) \in P_i} (x - x_G)^2 dx dy \quad (5.2)$$

où x_G désigne la coordonnée en abscisse du centre de gravité du contenant ($x_G = 0$). $d_{\min}(C_i, C_j)$ correspond à la distance euclidienne minimale entre les composants C_i et C_j . Cette dernière est calculée à partir des deux ensembles de segments des polygones de C_i et C_j . Pour effectuer ce calcul, une méthode naïve est utilisée. Un premier test vérifie si les deux ensembles de segments s'intersectent, dans une telle configuration la distance est nulle et le calcul s'arrête. Dans le cas contraire, la distance minimale entre chaque paire de segments est évaluée. Une représentation paramétrique des segments est alors utilisée pour effectuer ce calcul.

5.2.2 Premières optimisations

Avant d'analyser en détails les résultats obtenus pour la résolution de ce problème, deux optimisations sont réalisées. Une optimisation correspond à l'exécution complète de l'algorithme génétique avec la routine de séparation. Ces optimisations sont effectuées avec l'algorithme génétique *Omni-Optimizer*. Elles sont utilisées pour comprendre les mécanismes mis en jeu lors de la résolution du problème. Les paramètres par défaut de cet algorithme sont présentés dans la table 5.4. Les autres optimisations utiliseront cette base de paramètres. Les modifications de ces derniers seront spécifiées lors de chaque changement. On notera que contrairement à la plupart des problèmes résolus avec un algorithme génétique, les probabilités de croisement entre solutions sont très faibles (0.05 pour les variables réelles et 0 pour les variables discrètes). En effet, effectuer des croisements sur les variables de position entre deux solutions ne produit pas de bons individus dans un problème avec une forte compacité, même avec l'opérateur de sélection restreinte de l'algorithme *Omni-Optimizer*. En ce sens-là,

l'algorithme évolutionnaire utilisé est plus proche d'une stratégie évolutionnaire que d'un algorithme génétique.

La tolérance sur l'algorithme d'optimisation continue *BFGS* pour la séparation a été fixée à 10^{-8} . Son nombre d'itérations maximum a été fixé à 200 itérations. Nous avons constaté que la minimisation s'arrête bien avant, soit parce que les contraintes de placement sont respectées soit parce que la solution trouvée est un minimum local. Pour information, l'algorithme d'optimisation *BFGS* effectue en moyenne une vingtaine d'itérations.

Le nombre d'individus a été fixé à 100, de manière à avoir une bonne diversité des solutions initiales. Le nombre de générations a lui aussi été fixé à 100, après avoir constaté que l'hypervolume des surfaces de compromis n'évoluait guère après ce nombre. Ces constatations sont tirées des figures 5.22 et 5.24 qui sont présentées ci-après. Chaque optimisation évalue 10.000 solutions en 10 minutes dans un langage interprété.

Paramètres	Valeur
Nombre de générations	100
Nombre d'individus	100
Probabilité de croisement des variables réelles	0.05
Probabilité de mutation des variables réelles	0.4
Probabilité de croisement des variables discrètes/binaires	0
Probabilité de mutation des variables discrètes/binaires	0.3
Probabilité d'échange de deux composants d'une même solution	0.05
Dispersion pour l'opérateur de croisement continu η_c	5
Dispersion pour l'opérateur de mutation continu η_m	5
Coefficient relatif pour l'utilisation de l' ε -dominance	0.001
Utilisation du calcul des distances phénotypiques et génotypiques	Oui
Utilisation de l'opérateur de sélection restreinte	Oui

Table 5.4 – Paramètres de l'algorithme génétique utilisé

Pour évaluer les deux premières optimisations, le nombre de générations a toutefois été fixé à 200 pour visualiser l'évolution de la surface de compromis. La figure 5.4 présente les résultats des deux optimisations. Les sous-figures (a) et (m) présentent les surfaces de compromis obtenues. Les autres sous-figures présentent les solutions efficaces pour les deux optimisations. Les solutions sont triées dans l'ordre croissant du premier objectif f_1 . Il est intéressant de constater que l'ensemble des solutions proposées par chaque optimisation sont très similaires. Par la suite, on dira que deux solutions seront topologiquement différentes s'il n'est pas possible de passer de l'une à autre en appliquant des déplacements continus de ces composants. Même si les deux surfaces de compromis sont similaires, les solutions proposées par les deux optimisations sont complètement différentes. Cela montre l'influence de la population initiale et la diversité génotypique des solutions efficaces.

Avec des populations initiales aléatoires, l'algorithme génétique cherche d'abord des solutions réalisables. Une fois qu'une solution réalisable ou un ensemble de solutions réalisables a été trouvé, cet ensemble exerce une pression de sélection sur la population. Les figures 5.5 et 5.6 montrent l'évolution de la convergence de l'algorithme génétique. Chaque élément de ces figures représente une solution. Les croix représentent les solutions non réalisables, et les points les solutions réalisables. Les points cerclés représentent la surface de compromis pour une génération donnée. Les premières générations initialisées aléatoirement ne contiennent aucune solution réalisable. Il faut attendre la cinquième ou la dixième génération pour trouver une solution réalisable. Au bout de 30 générations, les deux populations contiennent uniquement des solutions réalisables. Comme l'algorithme est élitiste, plus aucune solution non réalisable ne sera reportée dans les générations à venir. Les sous-figures (p) de ces deux figures montrent l'évolution de l'hypervolume. Ce dernier a été normé à partir de l'hypervolume du front de Pareto, identifié à partir de l'ensemble des optimisations réalisées. Le point de référence \mathbf{W} utilisé pour évaluer les hypervolumes des surfaces de compromis a pour coordonnées $(5 \times 10^4, 0)$, et l'hypervolume de référence est évalué à 1.6635×10^6 . Pour les deux optimisations, sa progression ralentit fortement après 100 générations et stagne après 150 générations, signe que l'algorithme ne

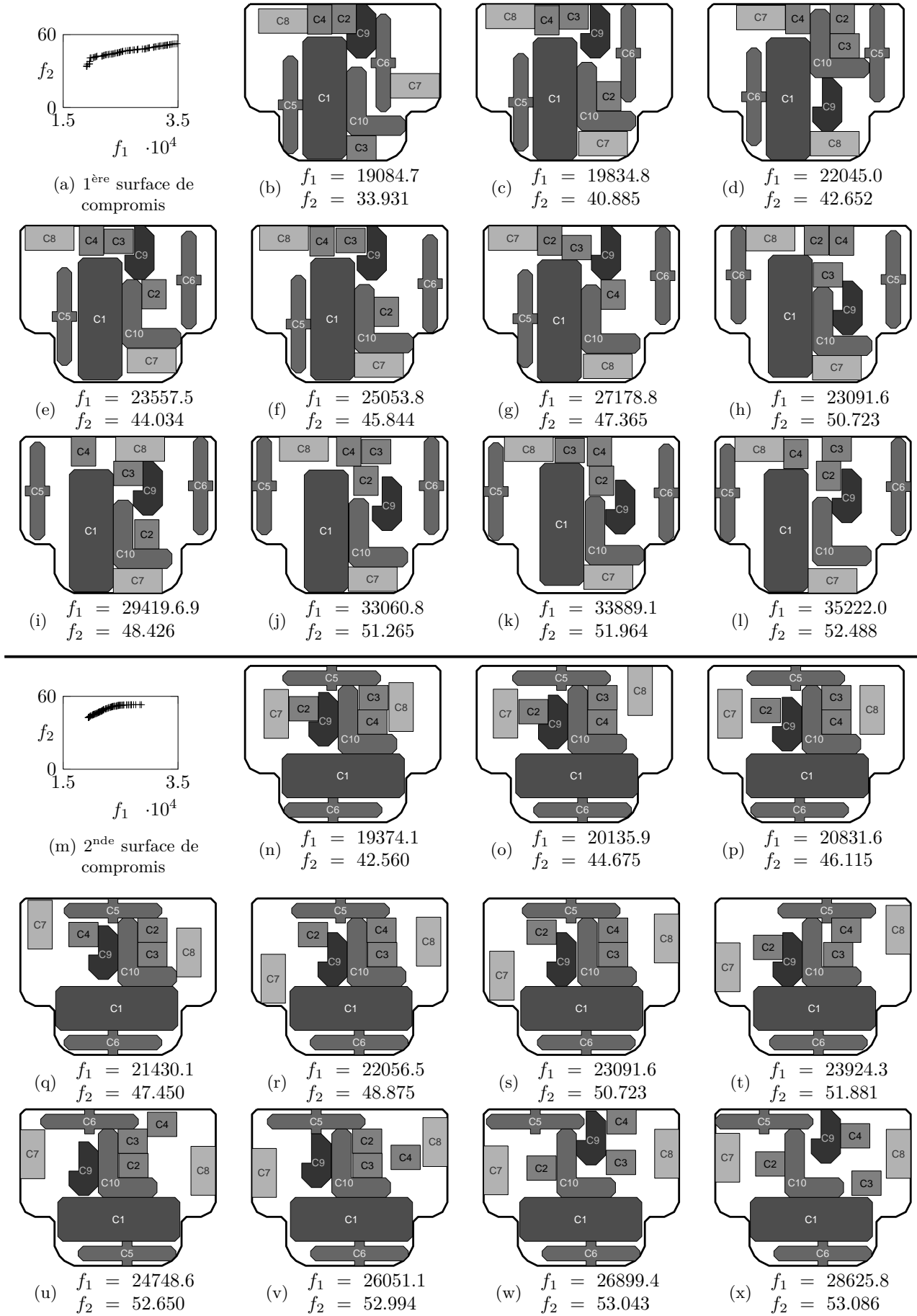


Figure 5.4 – Solutions extraites des deux surfaces de compromis de deux premières simulations. Les sous-figures (a) à (l) sont relatives à la 1^{ère} optimisation. Les sous-figures (m) à (x) sont relatives à la 2^{nde} optimisation.

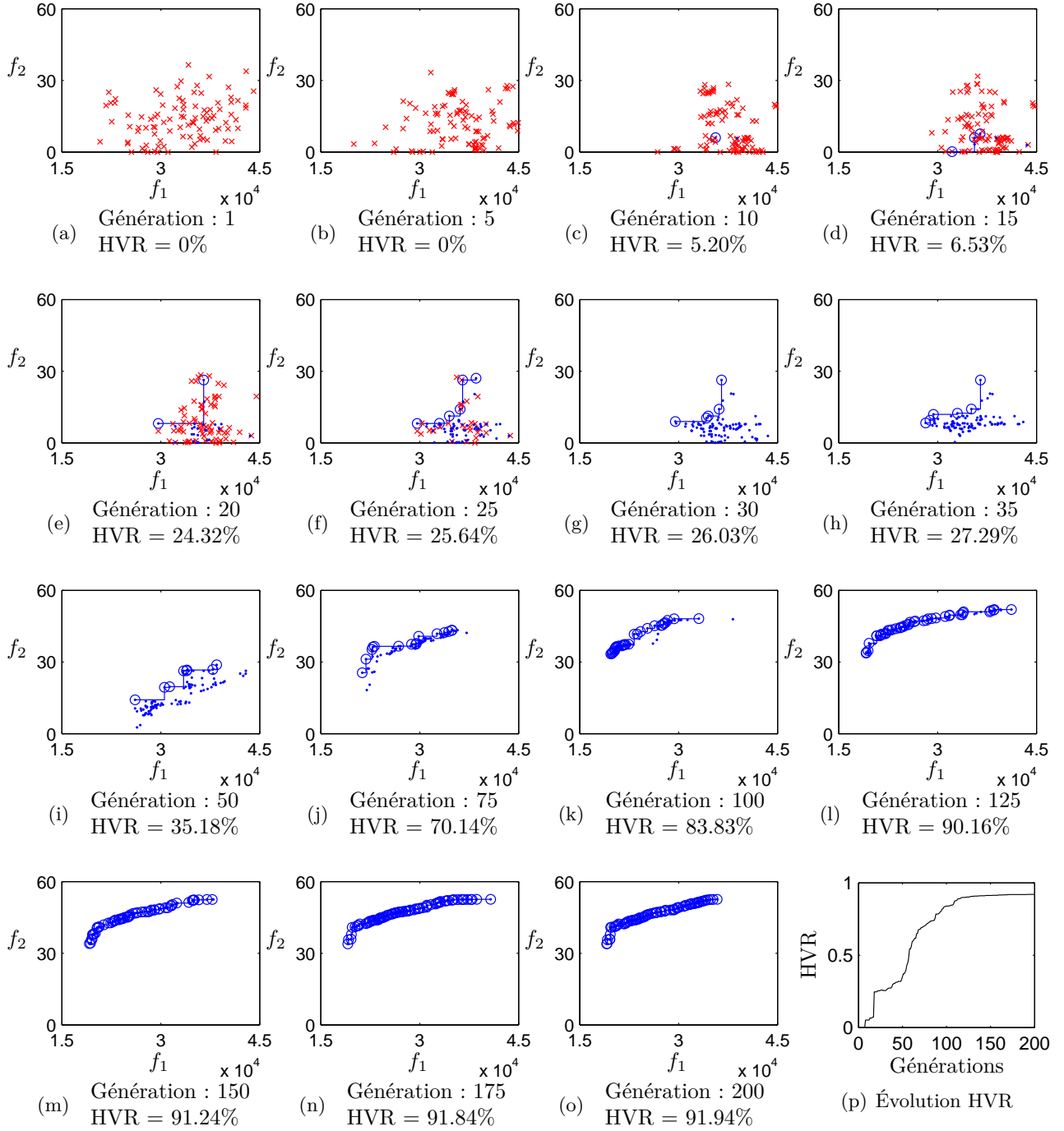


Figure 5.5 – Évolution de la population dans l'espace des objectifs pour la 1^{ère} optimisation. Les croix indiquent les solutions non réalisables, les points représentent les solutions réalisables. Les points non dominés sont cerclés. La sous-figure (p) montre l'évolution de l'hypervolume relatif au cours des générations.

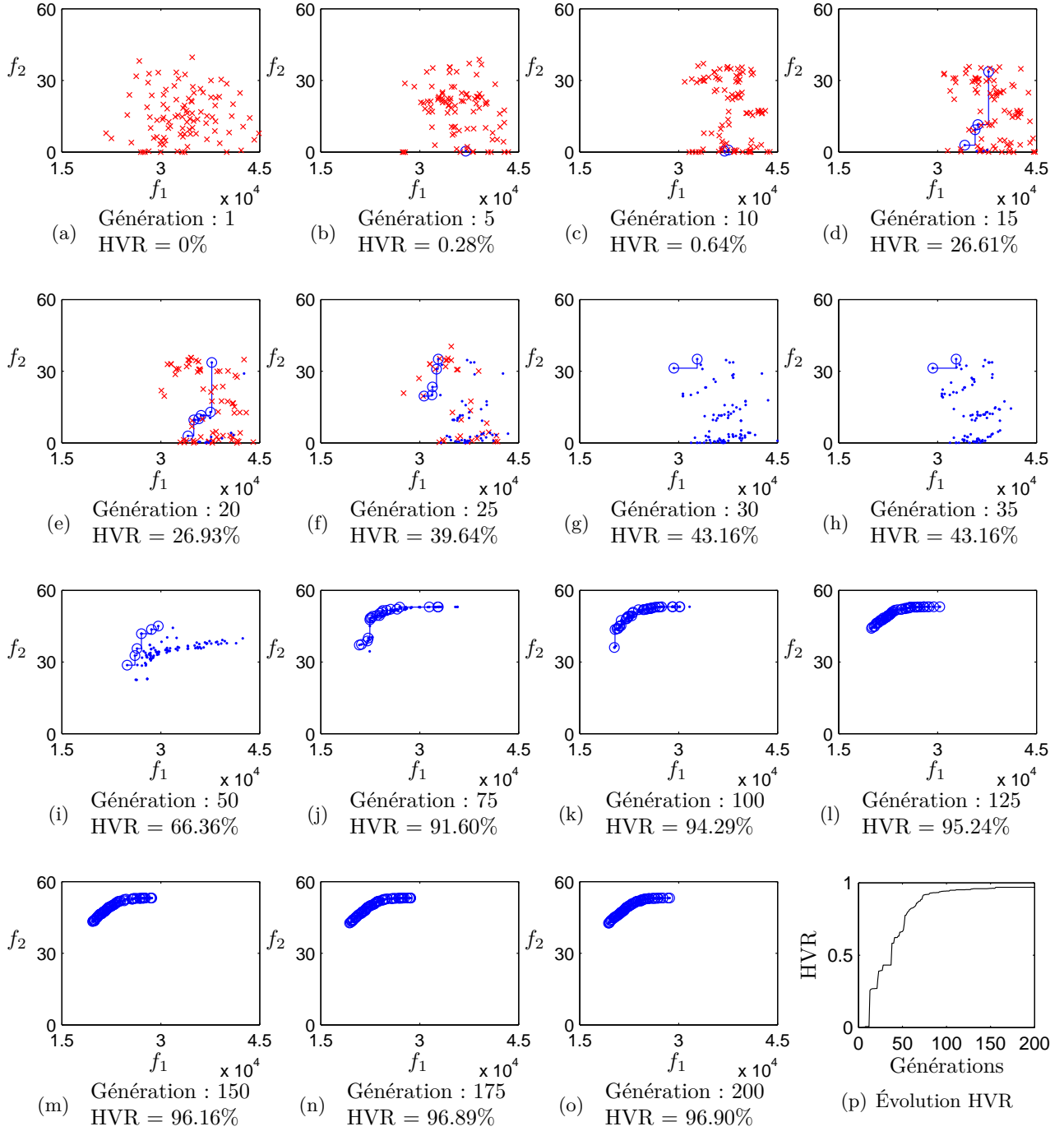


Figure 5.6 – Évolution de la population dans l'espace des objectifs pour la 2nde optimisation. Les croix indiquent les solutions non réalisables, les points représentent les solutions réalisables. Les points non dominés sont cerclés. La sous-figure (p) montre l'évolution de l'hypervolume relatif au cours des générations.

parvient plus à générer à de nouvelles solutions efficaces.

Pour comprendre pourquoi les solutions finales sont assez similaires, on se propose d'analyser les premières solutions réalisables obtenues par l'algorithme de placement. La figure 5.7 présente les premières solutions réalisables obtenues pour les deux premières optimisations. Pour la première optimisation, la sous-figure 5.7(b) est à l'origine de l'ensemble des solutions de la surface de compromis présentées sur les sous-figures 5.4(b) à 5.4(l). Pour la seconde optimisation, la sous-figure 5.7(f) semble être à l'origine de l'ensemble des solutions de la surface de compromis présentées sur les sous-figures 5.4(n) à 5.4(x). La comparaison des solutions des surfaces de compromis avec ces solutions montre que les solutions finales sont issues d'une seule et même solution réalisable identifiée lors des premières générations de l'algorithme.

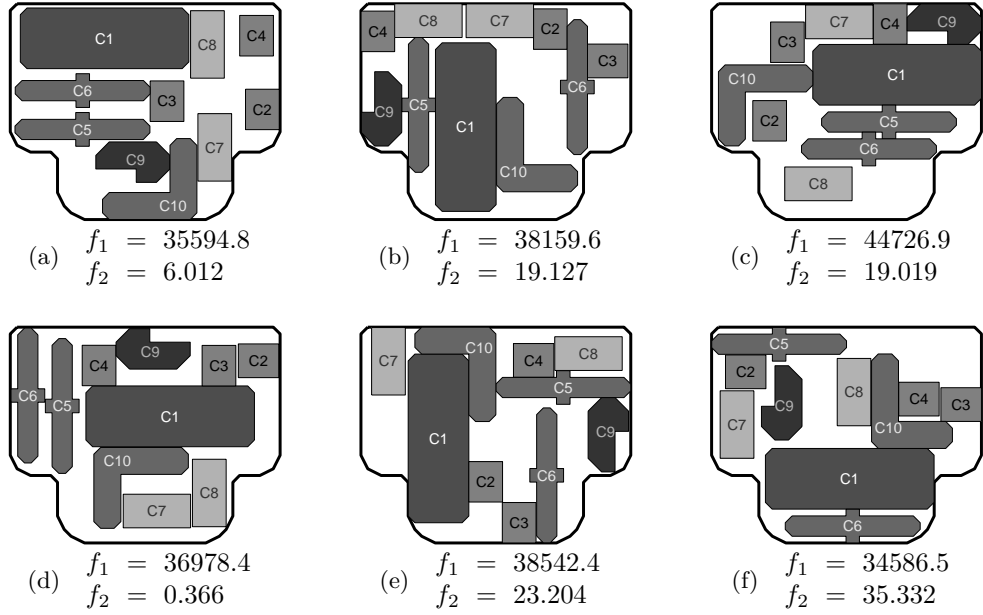


Figure 5.7 – Premières solutions réalisables identifiées pour les deux premières optimisations. Les sous-figures (a), (b), (c) sont issues de la première optimisation, les autres (d), (e), (f), de la seconde.

Pour analyser plus facilement les différences entre les solutions proposées par les deux optimisations, la méthode *Path-value* est utilisée pour visualiser les solutions [GDF73]. Elle permet de représenter les variables et objectifs d'une solution à l'aide d'une ligne brisée et de comparer plusieurs solutions facilement. La figure 5.8 représente les solutions efficaces identifiées lors des deux premières optimisations à l'aide de cette méthode. En abscisse sont représentées les variables réelles (1 à 20), les variables discrètes (21 à 30) et les objectifs (31 à 32). Les valeurs normalisées entre zéro et un des variables et objectifs sont en ordonnées. La normalisation des variables de position se fait à l'aide de leurs bornes. Les objectifs sont normés à l'aide du point idéal et du point Nadir du front de Pareto identifié. Ce front est présenté dans la section suivante. La sous-figure (a) correspond à la première optimisation, (b) à la seconde. Cette figure illustre le fait que les solutions des deux surfaces de compromis n'ont pas grand chose en commun, en revanche les solutions d'une même optimisation sont très similaires dans l'espace des variables.

La table 5.5 donne une idée de la répartition des tâches réalisées par le programme de résolution pour ces problèmes. Le temps total d'une optimisation est de 1160s pour évaluer 20.000 solutions. L'algorithme génétique représente seulement 2% du temps de calcul, soit un temps négligeable. L'évaluation des objectifs, qui fait intervenir différents calculs géométriques, occupe 15% du temps de calcul. L'algorithme de séparation, qui calcule les gradients analytiques pour l'optimisation continue, occupe les deux tiers du temps de calcul. Sans les formules analytiques des gradients, une différenciation numérique est nécessaire pour évaluer les gradients de la fonction F . Le temps de calcul d'une telle optimisation est de 13200s pour la résolution du même problème. Cela représente une multiplication par 11 du temps de calcul initial. Dans ce scénario, l'algorithme de séparation représente 97% du temps calcul. Cette comparaison montre que l'utilisation des gradients analytiques est essentielle.

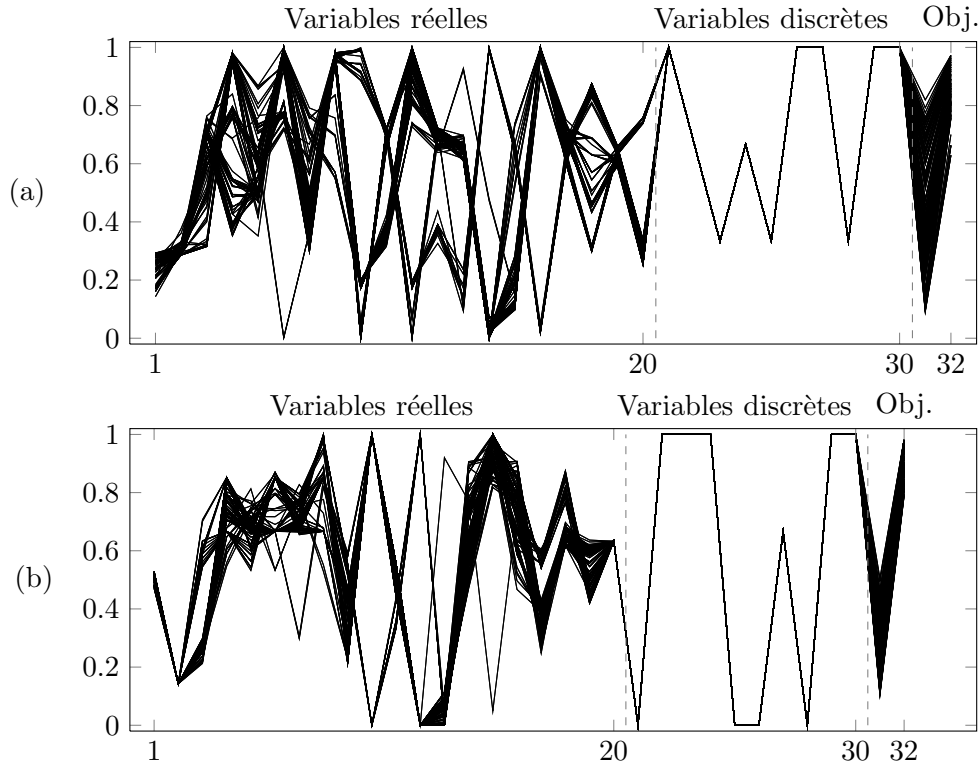


Figure 5.8 – Représentation des solutions efficaces à l'aide de la méthode *Path-value* pour les deux premières optimisations. La sous-figure (a) présente les solutions de la première optimisation, et (b) les solutions de la deuxième.

Tâches	Temps
Prétraitement	1%
Algorithme génétique	2%
Algorithme de séparation	66%
Évaluation des objectifs	15%
Post-traitement et autres	16%

Table 5.5 – Profilage du programme réalisé pour la résolution d'un problème d'agencement.

5.2.3 Solutions du front de Pareto

Le front de Pareto regroupe l'ensemble des points non dominés du problème. La complexité du problème et l'utilisation d'une méthode de résolution stochastique ne nous pas permettent pas de connaître exactement le front de Pareto. Toutefois, l'ensemble des optimisations réalisées nous permet de construire la meilleure approximation du front de Pareto. Par abus de langage, on omet le terme *approximation*, et on ne conserve uniquement l'expression *front de Pareto*. La figure 5.9 présente le front de Pareto du problème. Il a été obtenu à l'aide des différentes surfaces de compromis calculées à partir de l'ensemble des optimisations réalisées. Ce front a servi de référence pour les différents indicateurs multi-objectifs.

Sur le front de Pareto, des discontinuités peuvent être observées. Elles indiquent des changements de configurations des solutions efficaces. La figure 5.10 présente quelques-unes de solutions efficaces. La sous-figure (b) présente la solution extrême pour le premier objectif, qui minimise l'inertie transversale de l'assemblage. Sur cette solution, on observe que l'ensemble des composants sont centrés autour de l'axe de symétrie du coffre. La sous-figure (t) présente la seconde solution extrême, qui maximise les distances entre les composants C_5 , C_6 et C_7 , C_8 . On peut remarquer que les solutions (n) à (s) sont très proches de la valeur optimale de f_2 . En revanche, elles sont toutes bien meilleures en ce qui concerne l'objectif 1. Dans la pratique, un concepteur ne choisira pas la solution (t). En

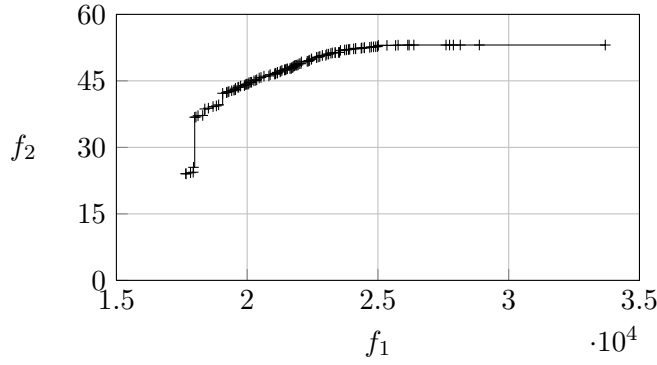


Figure 5.9 – Front de Pareto obtenu pour le premier problème d'agencement. Chaque point représente une solution efficace. En abscisse sont représentées les valeurs pour l'objectif f_1 , qui consiste à minimiser l'inertie transversale de l'assemblage. En ordonnées sont représentées les valeurs pour f_2 , qui consistent à maximiser les distances minimales entre les composants C_5 , C_6 ainsi que C_7 et C_8 .

dégradant le critère 2 de deux centièmes, il peut diminuer la valeur de l'inertie de 25%. Les solutions intermédiaires montrent les différentes configurations de solutions efficaces qui existent. Les solutions (e) à (j) sont très similaires et ont été obtenues lors d'une seule et même optimisation. Ces solutions proviennent de la seconde optimisation réalisée au début de cette section. Les solutions (l) et (m) montrent que deux solutions efficaces peuvent être très proches dans l'espace des objectifs, et être très différentes dans l'espace des variables. La figure 5.11 présente les solutions du front de Pareto à l'aide de la méthode *Path-value*. L'enchevêtrement des lignes brisées représentant les solutions efficaces du problèmes montre la diversité génotypique et phénotypique des solutions identifiées.

L'algorithme d'optimisation est conçu pour fournir l'ensemble des solutions efficaces. Clairement, il ne fournit pas toutes les solutions efficaces : même si le problème n'est pas entièrement symétrique par rapport à l'axe y , on peut envisager d'autres solutions symétriques par rapport à l'axe y . Pour avoir un problème complètement symétrique, il aurait fallu autoriser le retournement des composants, notamment pour le composant C_{10} de type T_4 . Le critère de distance génotypique de l'algorithme *Omni-Optimizer* intervient après le critère de dominance dans la sélection des solutions, ce qui explique son action soit si peu visible.

Davantage de diversité des solutions peut-être obtenue en regardant des solutions sous-optimales. Cette remarque soulève une question. Avec des ensembles de solutions tels que ceux présentés figure 5.4, le concepteur peut effectivement choisir de privilégier tel ou tel objectif. En revanche, il n'a pas le choix sur la topologie des solutions qui sont très similaires. La question que l'on peut se poser est donc la suivante : cherche-t-on uniquement les solutions efficaces quitte à perdre la diversité, ou cherche-t-on un ensemble de solutions quasi-optimales ? Quelques solutions dominées sont présentées figure 5.12. Ces solutions illustrent la diversité qui existe pour des solutions proches du front de Pareto.

Les sections suivantes analysent le rôle des différents composants de l'algorithme de résolution proposé.

5.2.4 Rôle de l'algorithme de séparation

Pour tester l'influence de l'algorithme de séparation, deux optimisations identiques sont exécutées. La première ne bénéficie pas de l'action de l'algorithme de séparation, la seconde fait appel à cet algorithme à chaque fois qu'une solution proposée ne satisfait pas les contraintes de placement. La figure 5.13 montre l'évolution statistique de l'indice de violation des contraintes de placement pour différentes générations. Les données de chaque génération sont résumées dans des boîtes à moustaches, contenant les valeurs minimales, médianes et maximales de l'indice de violation de contraintes de placement. Les premier et dernier quartiles sont aussi représentés. Sans la séparation, l'optimisation n'arrive pas à trouver une solution réalisable après 100 générations. Avec la séparation, tous les individus re-

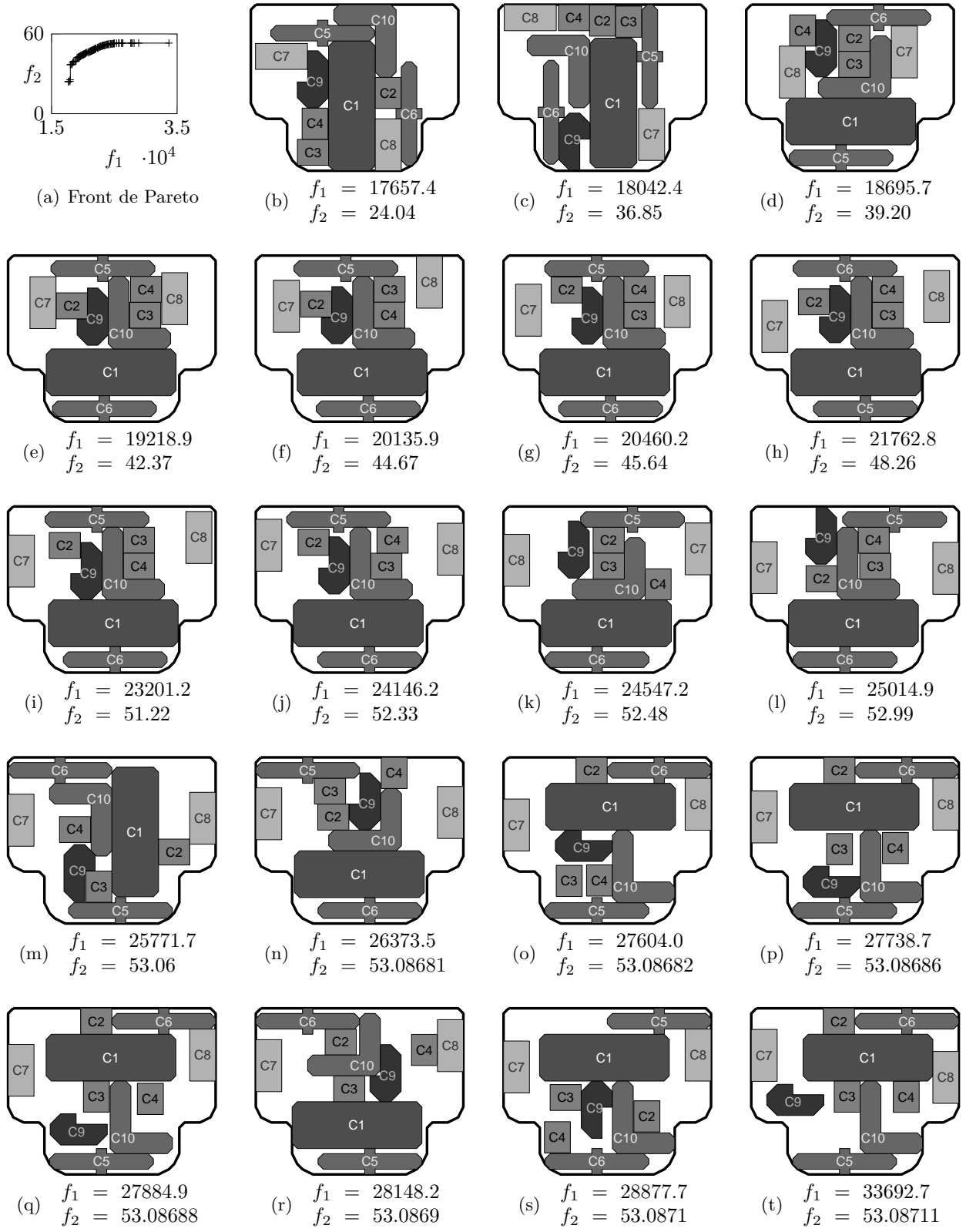


Figure 5.10 – Solutions extraites du front de Pareto.

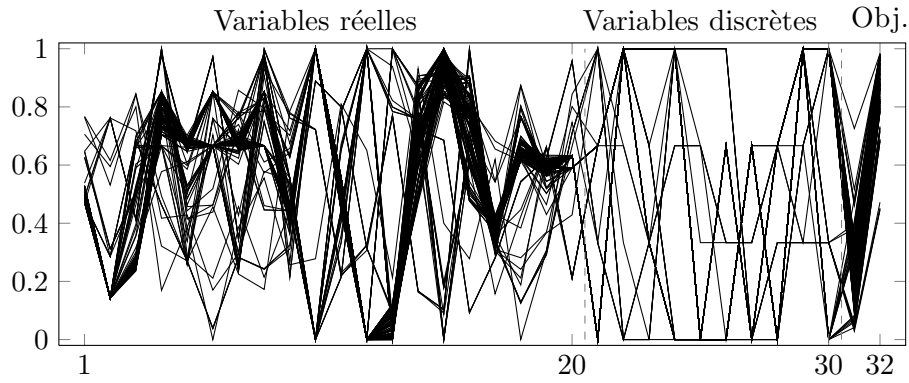


Figure 5.11 – Représentation des solutions efficaces du front de Pareto à l'aide de la méthode *Path-value*.

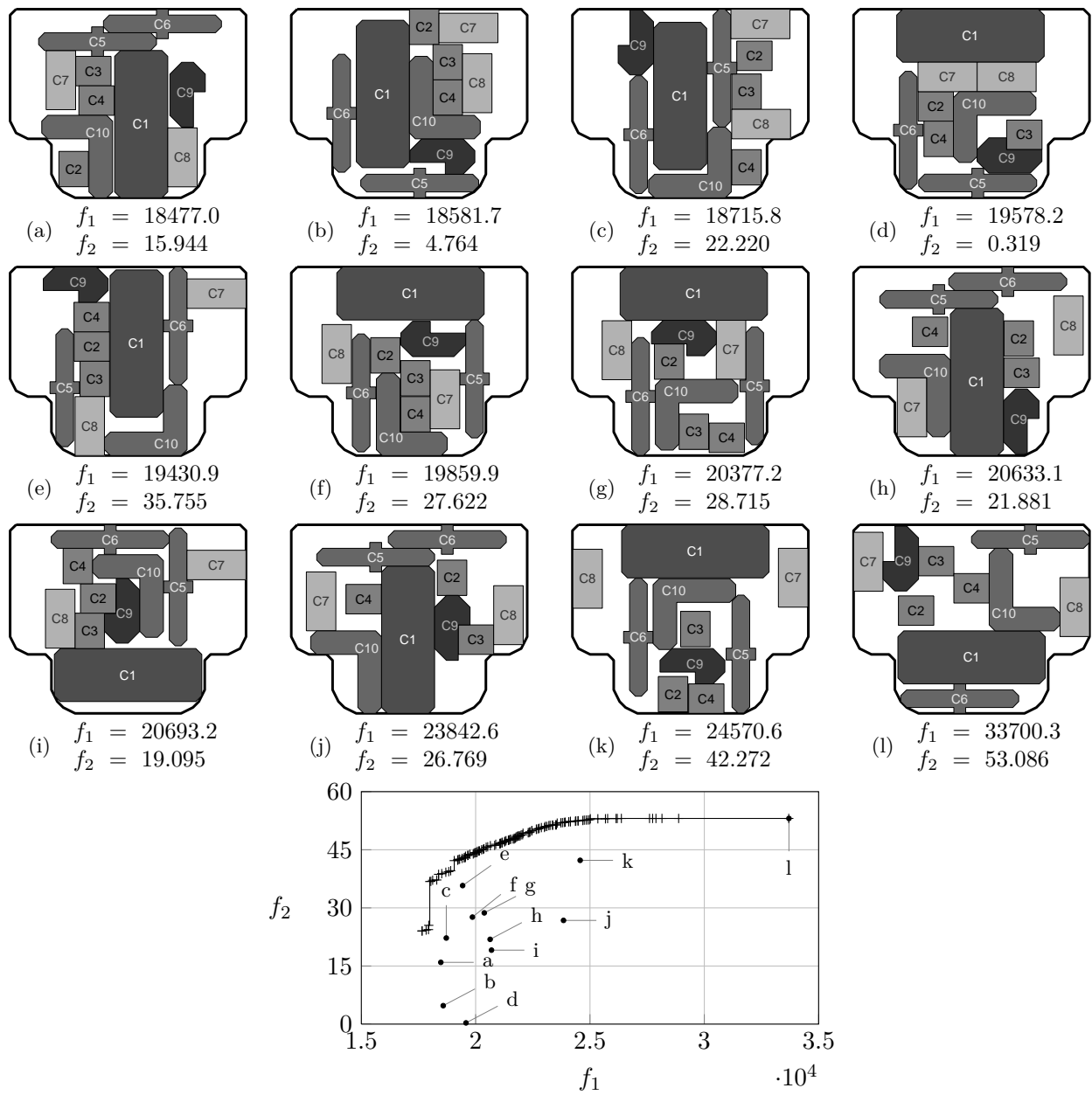


Figure 5.12 – Exemples de solutions sous-optimales. La sous-figure (m) rappelle le front de Pareto du problème, et place dans l'espace des objectifs les solutions sous-optimales présentées.

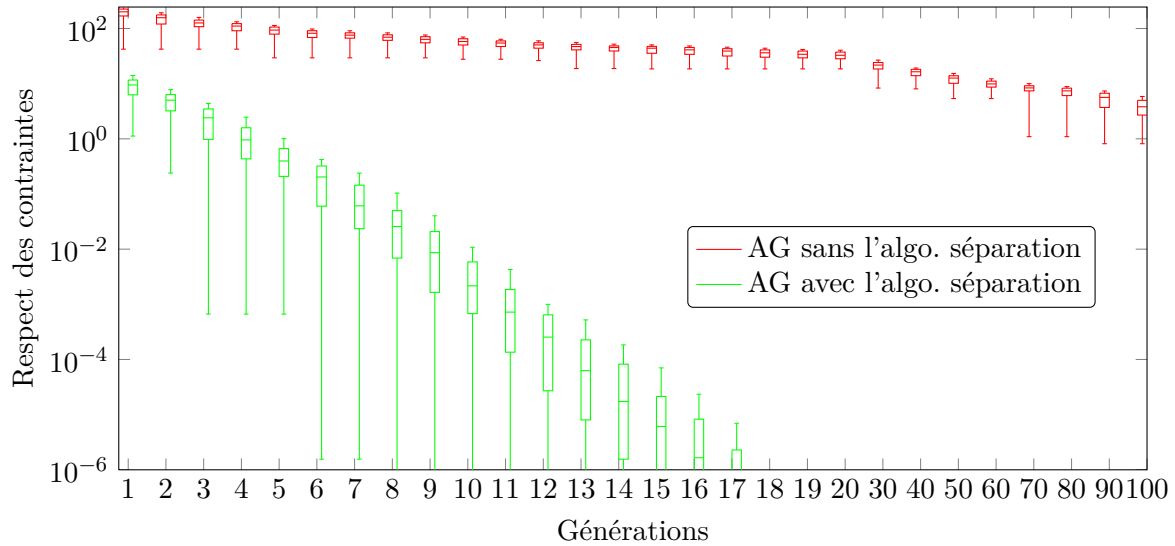


Figure 5.13 – Comparaison de l'évolution statistique du respect des contraintes de placement avec et sans l'algorithme de séparation. La différence entre ces deux simulations fait clairement apparaître le rôle de l'algorithme de séparation.

présentent des solutions réalisables au bout de 20 générations. La différence est visible dès la première génération, pour laquelle l'algorithme de séparation permet de diviser par cinq la valeur de l'indice de violation de contraintes. Une fois que l'ensemble des solutions est réalisable, l'algorithme d'optimisation peut réellement débiter l'optimisation multi-objectif. Cet exemple montre que l'algorithme d'optimisation globale a besoin de l'algorithme de séparation pour trouver des solutions réalisables et ensuite effectuer l'optimisation à proprement parler. Seul, l'algorithme d'optimisation fait diminuer l'indice de violation de contraintes, mais trop lentement pour voir apparaître des solutions réalisables en un nombre de générations raisonnables. Cette figure montre donc le rôle essentiel de l'algorithme de séparation.

5.2.5 Rôle des opérateurs génétiques

Les opérateurs génétiques ont pour objectif de générer de nouvelles solutions à partir d'une population parent. Pour mettre en évidence leurs rôles, on compare deux algorithmes : le premier est un algorithme génétique générationnel pseudo-aléatoire, le second correspond à l'algorithme présenté. Bénéficiant tous les deux de l'algorithme de séparation, la différence entre ces deux algorithmes se situe au niveau des opérateurs génétiques. Au lieu d'effectuer les modifications génétiques avec les paramètres classiques, l'algorithme pseudo-aléatoire mute chaque variable aléatoirement.

La comparaison entre les deux algorithmes s'effectue au niveau du respect des contraintes de placement à chaque génération. La figure 5.14 montre la vitesse de convergence de l'indice de violation de contraintes, qui exprime le non-respect des contraintes de placement. Pour chaque génération, les données statistiques de l'indice de violation de contraintes (médiane, 1^{er} et 3^{ème} quartiles, minimum et maximum) sont représentées à l'aide de boîtes à moustaches. Les populations initiales sont choisies identiques, les premières boîtes à moustaches sont donc identiques. L'utilisation de l'élitisme empêche de dégrader l'ensemble des solutions entre deux générations, ce qui explique le caractère strictement décroissant des différentes statistiques. La convergence de l'algorithme est plus rapide avec les bons opérateurs génétiques. Avec l'algorithme proposé, l'ensemble des individus respectent les contraintes de placement au bout de six générations, alors qu'avec l'algorithme pseudo-aléatoire, il faut attendre vingt générations. Les surfaces de compromis de ces deux optimisations permettent de conclure la même chose : l'utilisation des *vrais* opérateurs génétiques permet d'obtenir de meilleurs résultats. La figure 5.20, présentée ci-après, illustre ces différences sur les surfaces de compromis. La différence entre ces résultats va s'accroître avec des problèmes à plus forte compacité, comme dans le prochain exemple.

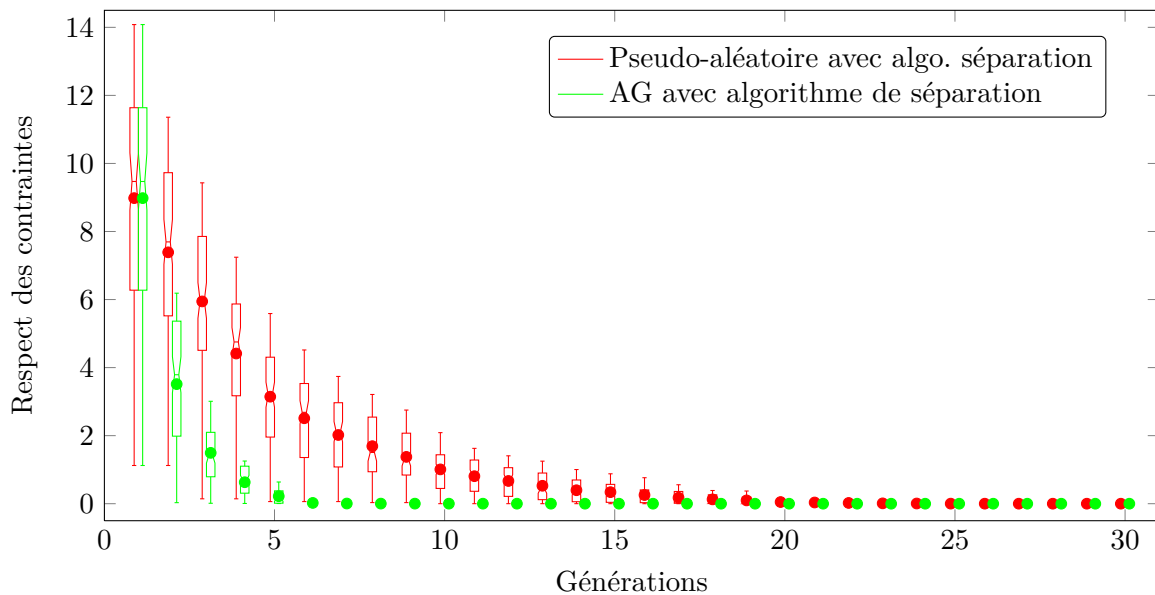


Figure 5.14 – Comparaison de l'évolution du respect des contraintes de placement pour un algorithme pseudo-aléatoire et un algorithme génétique. Les deux algorithmes utilisent la procédure de séparation. Pour chaque génération, les données statistiques (médiane, 1^{er} et 3^{ème} quartiles, minimum et maximum) sont représentées à l'aide de boîtes à moustaches. La moyenne des échantillons est représentée à l'aide d'un point. La figure montre les résultats obtenus pour une même population initiale, par conséquent les boîtes à moustaches de la première génération sont identiques.

5.2.6 Réglage des opérateurs génétiques

5.2.6.1 Influence de la probabilité de croisement p_c

Tout d'abord, l'influence de la probabilité de croisement sur les variables réelles est discutée. Dans les réglages initiaux de l'algorithme génétique (Tab. 5.4), une valeur très faible de cette probabilité est indiquée. La justification vient des différentes optimisations initiales réalisées. La figure 5.15 montre l'influence de l'opérateur de croisement sur la convergence de l'algorithme. Sur cet exemple, différentes optimisations sont réalisées avec la même population initiale. La probabilité de mutation des variables réelles est fixée à 30%, la probabilité d'échange est de 10%. La probabilité de croisement des variables réelles p_c est échantillonnée tous les 10%. Les meilleures surfaces de compromis sont obtenues pour $p_c = 0\%$ et $p_c = 10\%$, soit des probabilités faibles. Ceci peut s'expliquer par le fait que les croisements introduisent trop de changements dans les nouvelles solutions, qui ont peu de chance d'être meilleures que leurs parents et donc d'être sélectionnées pour les générations suivantes. Par la suite, une probabilité de croisement sur les variables réelles de 5% sera utilisée.

5.2.6.2 Influence des probabilités de mutation p_m et d'échange p_s

Pour analyser l'influence des probabilités d'intervention des opérateurs génétiques de mutation et d'échange, quatre simulations sont élaborées. Une simulation correspond à un ensemble de paramètres avec une variation sur un paramètre donné. La table 5.6 présente les différentes simulations effectuées. Pour chaque simulation, la valeur du paramètre variant évolue entre 0% et 100% avec un pas de 10%.

Simulations	Type de variations	Autres réglages	
\mathcal{S}_1	Variation sur l'opérateur de mutation	Probabilité d'échange	$p_s = 0\%$
\mathcal{S}_2	Variation sur l'opérateur de mutation	Probabilité d'échange	$p_s = 30\%$
\mathcal{S}_3	Variation sur l'opérateur d'échange	Probabilité de mutation	$p_m = 0\%$
\mathcal{S}_4	Variation sur l'opérateur d'échange	Probabilité de mutation	$p_m = 30\%$

Table 5.6 – Récapitulatif des différentes simulations.

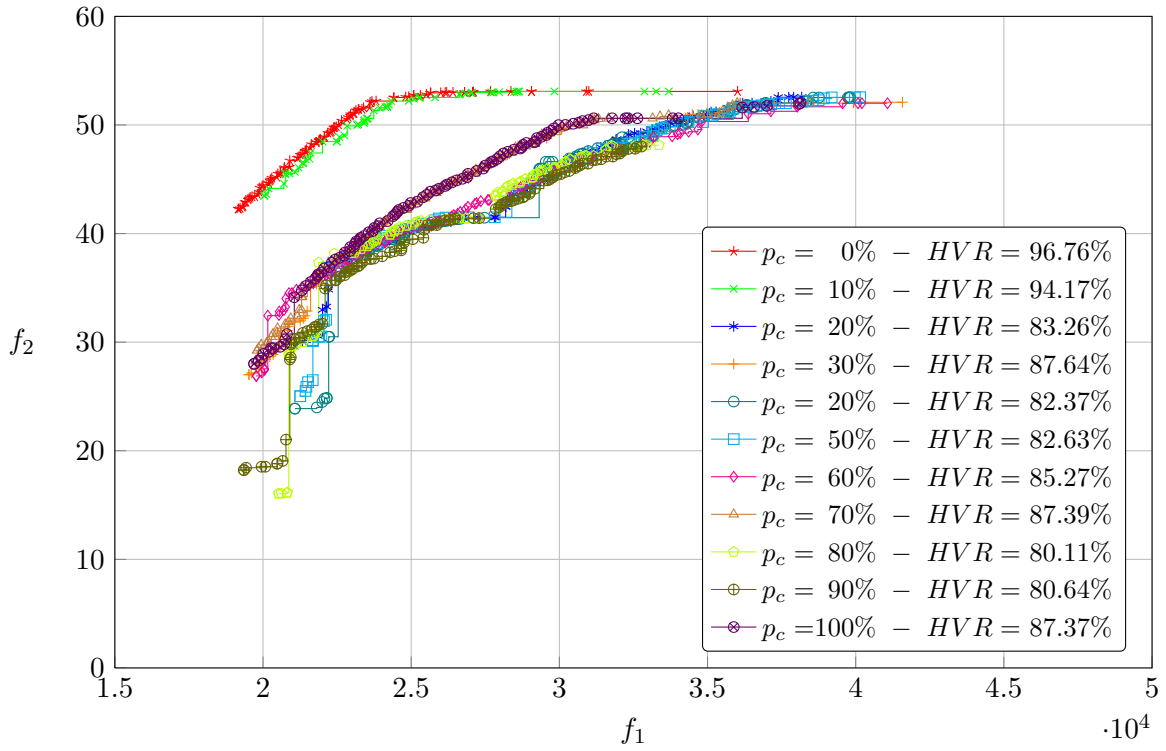


Figure 5.15 – Surfaces de compromis obtenues pour différentes probabilités de croisement p_c . Le paramètre de mutation réelle est réglé à 30%. ($p_m = 30\%$).

Les algorithmes utilisés sont stochastiques, cela signifie que chaque optimisation doit être effectuée plusieurs fois avec des populations initiales différentes avant de pouvoir tirer des conclusions. Une simulation regroupera plusieurs optimisations partageant les mêmes réglages mais une population initiale différente. Pour chaque combinaison de paramètres de chaque simulation, six optimisations sont effectuées avec des populations initiales différentes. D'une simulation à l'autre et d'une variation à l'autre, les mêmes populations initiales sont utilisées, ce qui permet d'écarter le rôle de la population initiale lors de la comparaison de deux optimisations. Au total, $4 \times 11 \times 6 = 264$ optimisations sont réalisées, où 4 représente le nombre de simulations, 11 le nombre de variations par simulation et 6 le nombre d'optimisations par variation.

Les figures 5.16, 5.17, 5.18 et 5.19 présentent une étude comparative sur l'influence des interventions des opérateurs génétiques. Chaque surface de compromis, présentée sur ces figures, est obtenue en prenant les meilleures solutions des six optimisations réalisées pour chaque variation de paramètre. Le fait de comparer les résultats sur plusieurs optimisations permet d'atténuer le rôle de la population initiale. Comparer deux surfaces de compromis revient à comparer les meilleures solutions obtenues par deux ensembles d'optimisations ayant des paramètres de réglages différents et des populations initiales identiques. Les différences observées entre les surfaces sont donc directement reliées aux différences de probabilités d'action des opérateurs génétiques. Si on avait choisi un nombre d'optimisations plus grand pour chaque variation, les différences entre les surfaces de compromis auraient été encore plus atténuées, et les résultats auraient eu moins de sens. Pour chaque surface de compromis, l'hypervolume relatif est présenté dans la légende.

La figure 5.20 présente les surfaces de compromis extraites pour chacune des simulations \mathcal{S}_1 , \mathcal{S}_2 , \mathcal{S}_3 et \mathcal{S}_4 . Ces surfaces ont été obtenues en sélectionnant les meilleures solutions obtenues lors d'une simulation (66 optimisations). En analysant les surfaces de compromis sur autant d'optimisations, les différences sont encore plus atténuées, et chaque surface de compromis s'approche davantage du front de Pareto. Cette figure met en avant le fait que la simulation \mathcal{S}_3 n'arrive pas à obtenir de bons résultats. À ces différentes surfaces de compromis, a été ajoutée la surface de compromis obtenue avec 200 optimisations de l'algorithme pseudo-aléatoire élitiste générationnel. Ce dernier a réussi à identifier

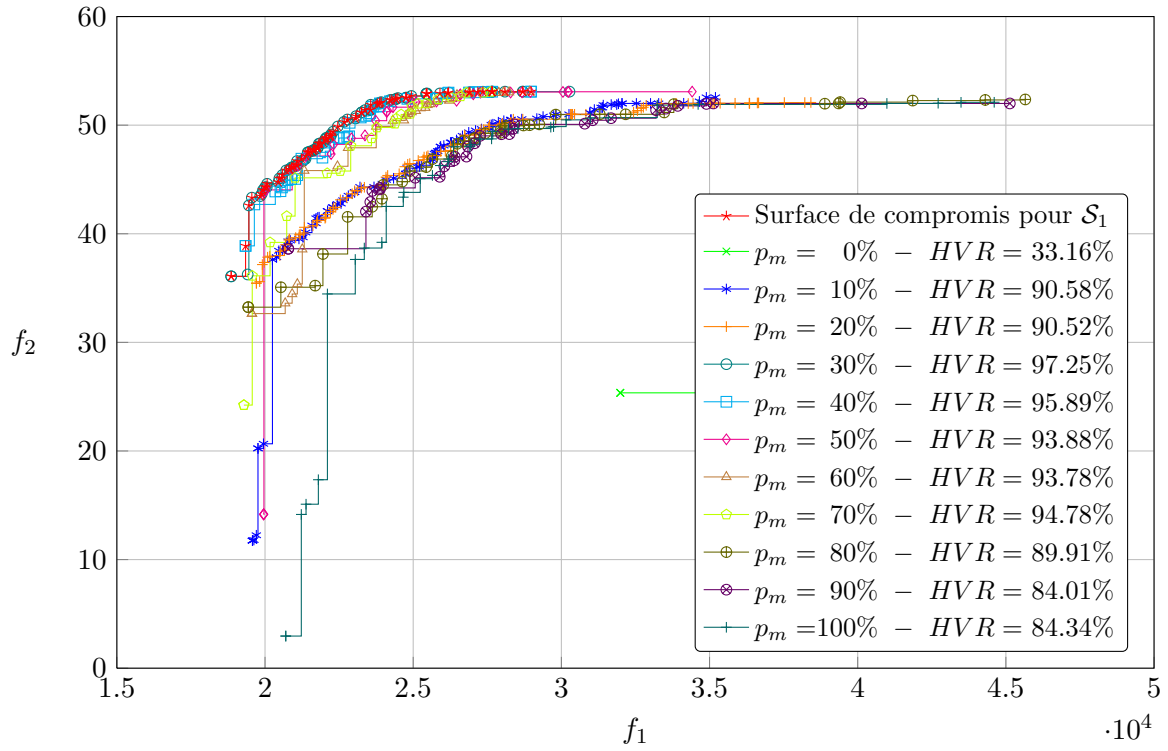


Figure 5.16 – Surfaces de compromis de S_1 : Influence de la probabilité de mutation réelle avec une probabilité d'échange des composants nulle. ($p_s = 0.0\%$).

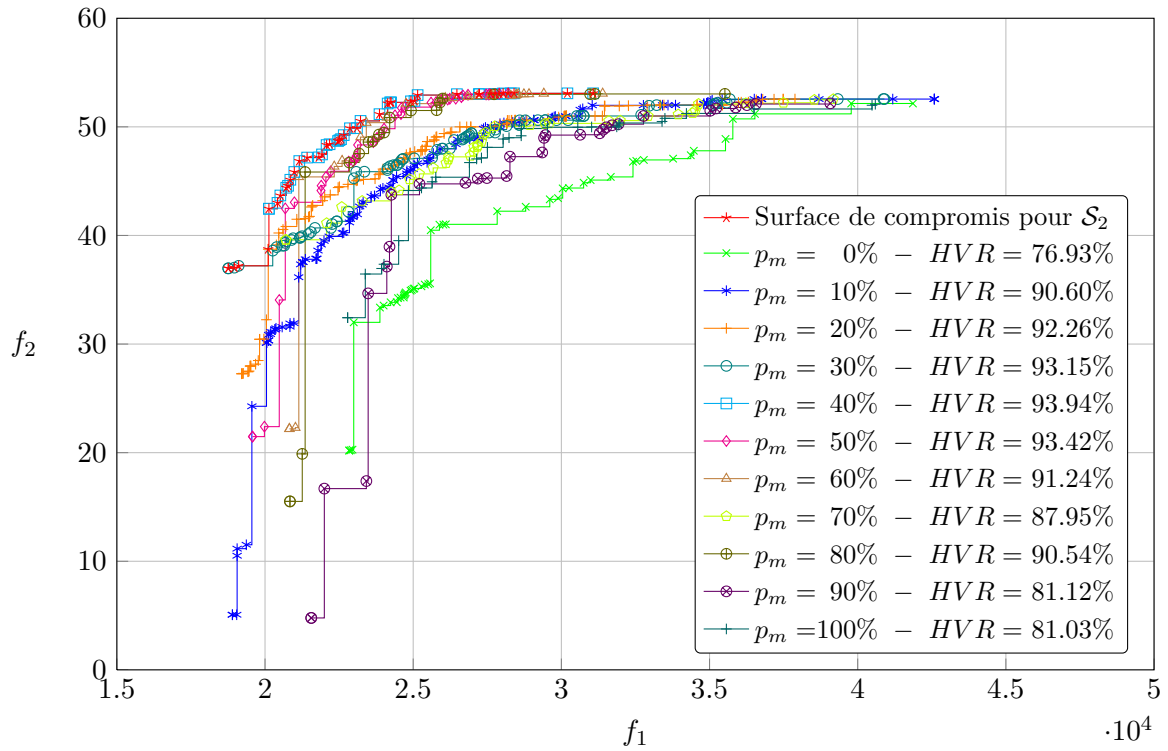


Figure 5.17 – Surfaces de compromis de S_2 : Influence de la probabilité de mutation réelle avec une probabilité d'échange des composants de 30.0%. ($p_s = 30.0\%$).

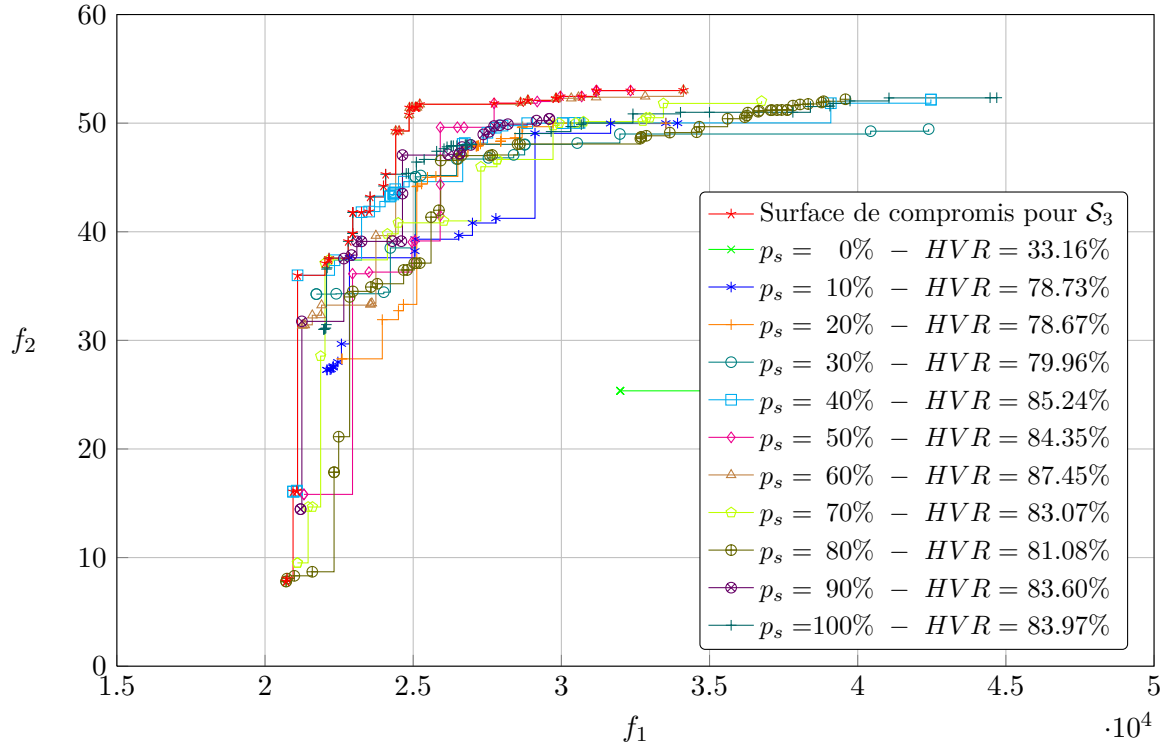


Figure 5.18 – Surfaces de compromis de \mathcal{S}_3 : Influence de la probabilité d'échange avec une probabilité de mutation sur les variables réelles nulle. ($p_m = 0.0\%$).

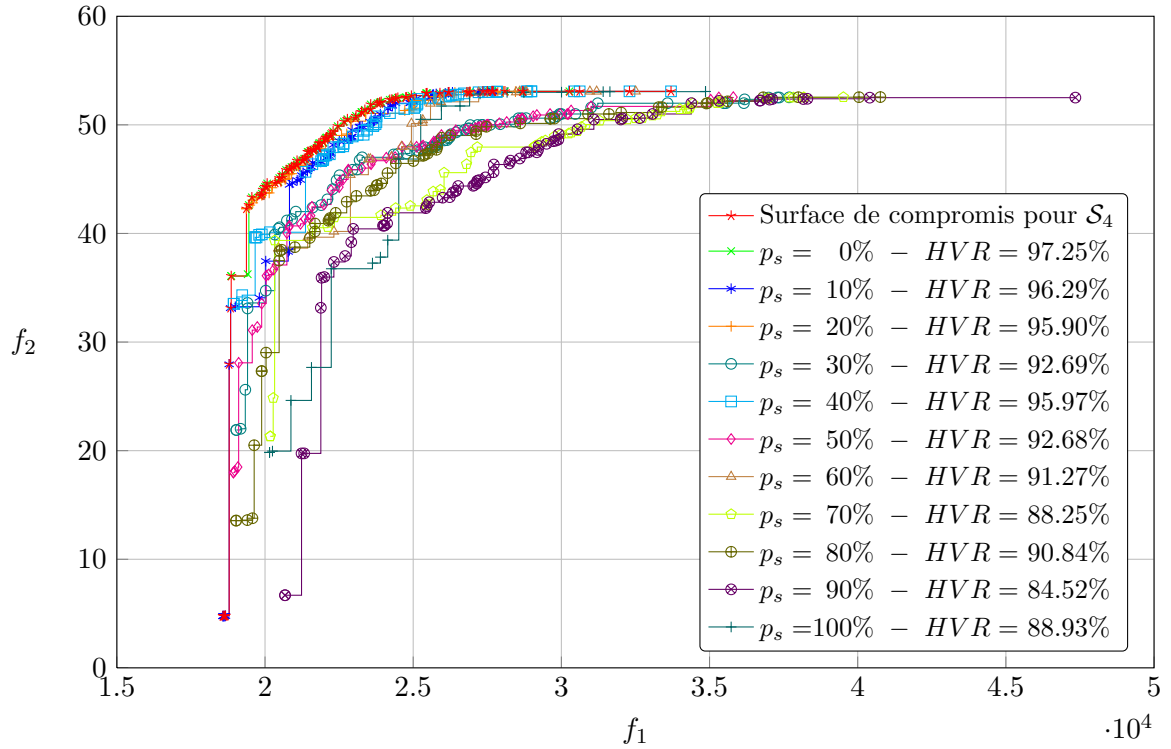


Figure 5.19 – Surfaces de compromis de \mathcal{S}_4 : Influence de la probabilité d'échange avec une probabilité de mutation sur les variables réelle de 30.0%. ($p_m = 30.0\%$).

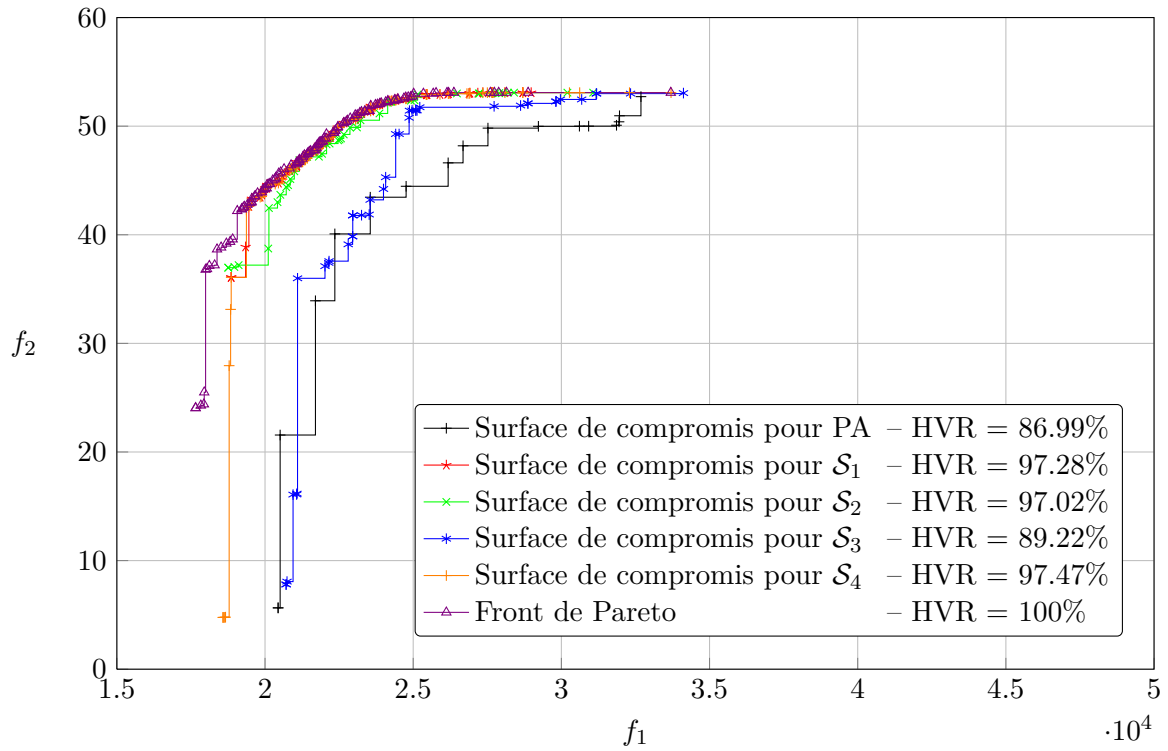


Figure 5.20 – Comparaison des différentes surfaces de compromis extraites des simulations \mathcal{S}_1 , \mathcal{S}_2 , \mathcal{S}_3 et \mathcal{S}_4 avec le front de Pareto. Le terme PA de la légende fait référence à l’algorithme pseudo-aléatoire bénéficiant de l’algorithme de séparation.

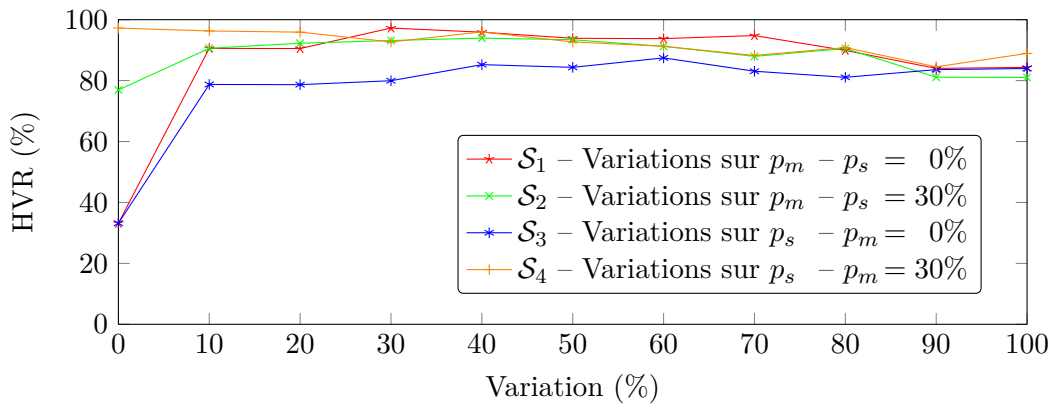


Figure 5.21 – Comparaison des valeurs de l’hypervolume relatif HVR pour les différentes simulations \mathcal{S}_1 , \mathcal{S}_2 , \mathcal{S}_3 et \mathcal{S}_4 .

	Variation	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
\mathcal{S}_1	HVR (%)	33.16	90.58	90.52	97.25	95.89	93.88	93.78	94.78	89.91	84.01	84.34
\mathcal{S}_2	HVR (%)	76.93	90.60	92.26	93.15	93.94	93.42	91.24	87.95	90.54	81.12	81.03
\mathcal{S}_3	HVR (%)	33.16	78.73	78.67	79.96	85.24	84.35	87.45	83.07	81.08	83.60	83.97
\mathcal{S}_4	HVR (%)	97.25	96.29	95.90	92.69	95.97	92.68	91.27	88.25	90.84	84.52	88.93

Table 5.7 – Récapitulatif des valeurs obtenues pour l'hypervolume relatif pour les différentes simulations. Chaque valeur présente le HVR obtenu pour six optimisations avec les mêmes paramètres. Les cellules grisées présentent les meilleures valeurs pour chaque simulation. Pour comparaison, les 200 optimisations réalisées avec l'algorithme pseudo-aléatoire ont permis d'obtenir un HVR de 86.99%.

	\mathcal{S}_1	\mathcal{S}_2	\mathcal{S}_3	\mathcal{S}_4
\mathcal{S}_1	0	0.7381	1	0.3100
\mathcal{S}_2	0.1294	0	1	0.1800
\mathcal{S}_3	0.0353	0	0	0.0100
\mathcal{S}_4	0.2706	0.7381	1	0

Table 5.8 – Comparaison des surfaces de compromis de \mathcal{S}_1 à \mathcal{S}_4 à l'aide de la métrique \mathcal{C} .

trois solutions dominant la surface de compromis de \mathcal{S}_3 , pour laquelle l'opérateur de mutation réelle est inexistant. Cela montre l'importance du rôle de cet opérateur ainsi que l'importance des réglages des paramètres de l'algorithme d'optimisation.

La table 5.7 résume les hypervolumes relatifs des surfaces de compromis pour chaque simulation et chaque variation. La figure 5.21 présente graphiquement ces résultats. Les points qui se superposent, correspondent à des calculs identiques. La table 5.8 présente une comparaison des meilleures surfaces de compromis extraites des simulations \mathcal{S}_1 à \mathcal{S}_4 à l'aide de la métrique \mathcal{C} présentée section 2.4.2.1. Ces résultats viennent compléter ceux de la figure 5.20. Cette figure et cette table montrent que la simulation \mathcal{S}_3 faisant varier la probabilité d'échange avec une probabilité de mutation nulle n'a jamais obtenu de bons résultats. Cette simulation basée uniquement sur les opérateurs de croisement et d'échange montre l'importance de l'opérateur de mutation, dont les autres simulations ont bénéficié. Les tables C.1, C.2, C.3 et C.4 présentent en annexe page 163 les statistiques de l'hypervolume relatif pour les différentes optimisations réalisées à partir des quatre simulations \mathcal{S}_1 , \mathcal{S}_2 , \mathcal{S}_3 et \mathcal{S}_4 .

La figure 5.22 présente une comparaison de l'évolution statistique de l'hypervolume relatif pour les simulations \mathcal{S}_1 , \mathcal{S}_2 , \mathcal{S}_3 et \mathcal{S}_4 et à différentes étapes de la convergence. La comparaison statistique est résumée à l'aide de boîtes à moustaches. L'objectif est de faire tendre la valeur de l'hypervolume relatif vers un. Sur la première colonne de cette figure, les simulations \mathcal{S}_1 et \mathcal{S}_2 sont comparées deux à deux. Les boîtes à moustaches de gauche représentent l'évolution de l'hypervolume pour les simulations \mathcal{S}_1 , celles de droites les simulations \mathcal{S}_2 . Sur la seconde colonne de cette figure, les simulations \mathcal{S}_3 et \mathcal{S}_4 sont comparées deux à deux. Les boîtes à moustaches de gauche représentent l'évolution de l'hypervolume pour les simulations \mathcal{S}_3 , celles de droites les simulations \mathcal{S}_4 . En parcourant verticalement les sous-figures, la probabilité de mutation (1^{ère} colonne) et la probabilité d'échange (2^{nde} colonne) augmentent.

La première conclusion à tirer de ces sous-figures est qu'aucune solution réalisable n'a été trouvée dans les populations initiales (HVR=0). Il faut attendre cinq ou dix générations avant d'identifier des solutions réalisables. Cette caractéristique s'accroît dès lors que la compacité du problème augmente. Tant qu'aucune solution ne respecte les contraintes de placement, l'algorithme génétique se comporte comme un algorithme mono-objectif. En effet, le classement des solutions se fait d'abord sur l'indice de violation des contraintes. Si cet indice n'est pas nul, alors le rang des solutions est déterminé à partir du rang de ces indices. Il est donc inutile de parler d'optimisation multi-objectif tant que les solutions ne sont pas réalisables. Pour accélérer la convergence et assurer une certaine diversité génotypique des solutions, il faut initialiser les solutions à l'aide d'heuristiques de placement ou bien utiliser des solutions fournies par l'utilisateur.

Sur les sous-figures (a) et (b), les simulations \mathcal{S}_1 et \mathcal{S}_3 sont identiques (Paramètres AG identiques, populations initiales identiques). Les résultats pour ces deux simulations sont donc identiques. La faible probabilité d'action des opérateurs génétiques rend difficile la convergence. Toutefois, l'hybridation de

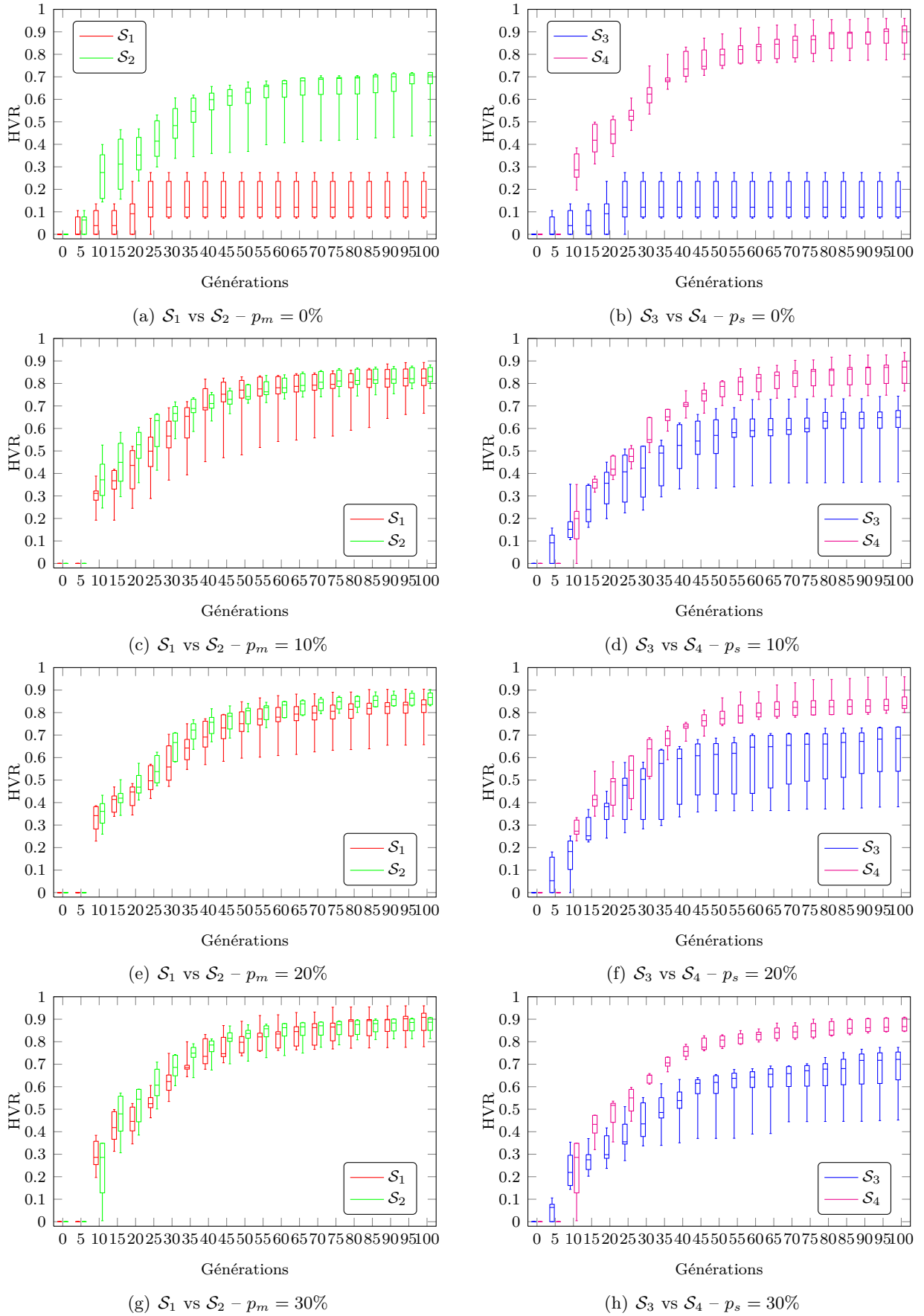


Figure 5.22 – Comparaison statistique de l'évolution des valeurs de l'hypervolume relatif pour les différentes simulations à différentes générations. \mathcal{S}_1 & \mathcal{S}_2 : Influence de l'opérateur de mutation avec une probabilité respective d'échange de 0% et 30%. \mathcal{S}_3 & \mathcal{S}_4 : Influence de l'opérateur d'échange avec une probabilité respective de mutation de 0% et 30%.

l'algorithme génétique et l'algorithme de séparation permet d'identifier au moins une solution réalisable à chaque exécution. Cette solution étant bien souvent unique, l'algorithme d'optimisation n'arrive pas à générer de nouvelles solutions efficaces. Les sous-figures (a), (c), (e) et (g) montrent que les valeurs d'hypervolume sont moins dispersées avec l'opérateur d'échange. Cela signifie que l'hypervolume des solutions évoluent de manière plus régulière et cohérente. Sans cet opérateur, une simulation peut stagner et ne pas arriver à produire de meilleures solutions. Les sous-figures (b), (d), (f) et (h) montrent l'importance du rôle de l'opérateur de mutation. Sans lui, les surfaces de compromis atteignent au maximum 70% de l'hypervolume du front de Pareto.

La figure 5.22 permet d'établir plusieurs conclusions. Sur la colonne de gauche, on observe que l'opérateur d'échange permet d'accélérer la convergence initiale. Toutefois, il ne permet pas d'améliorer la qualité des surfaces de compromis. La colonne de droite montre clairement que l'opérateur de mutation joue un rôle crucial dans la convergence du problème.

5.2.7 Choix de l'algorithme d'optimisation globale

Initialement, nous avons choisi l'algorithme *Omni-Optimizer* pour effectuer l'optimisation globale. Ce choix était motivé par les caractéristiques intéressantes de ce nouvel algorithme. Nous nous proposons de comparer ici les performances de cet algorithme avec *NSGA-II*.

Les comparaisons sur ces deux algorithmes très proches permettent de mettre en avant les avantages de chacun. Leurs différences de comportement sont analysées par le biais des différentes métriques présentées auparavant (Section 2.4). Les deux algorithmes sont testés sur une simulation, où la probabilité de mutation des variables réelles évolue de 10% à 60% avec un pas de 10%. La probabilité d'échange de deux composants est de 10%. Sur ces six variations, vingt optimisations sont effectuées avec 100 individus sur 200 générations. En tout, ce sont 240 optimisations qui ont été effectuées, soit 4.8×10^6 solutions évaluées. Ces résultats sont analysés dans les différentes tables et figures présentées ci-après.

5.2.7.1 Comparaison sur l'hypervolume relatif

La figure 5.23 présente les résultats comparatifs entre les deux algorithmes de référence pour l'hypervolume relatif. Les résultats apparaissent sensiblement meilleurs pour l'algorithme *Omni-Optimizer*. La table C.9, présentée en annexe page 166, résume numériquement les résultats statistiques l'hypervolume relatif pour les deux algorithmes.

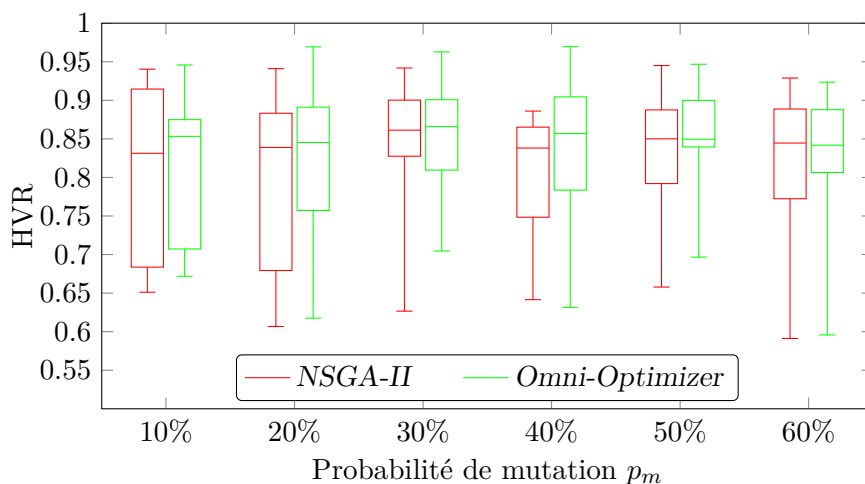


Figure 5.23 – Comparaison des hypervolumes relatifs pour les algorithmes *NSGA-II* et *Omni-Optimizer*. Pour chaque probabilité de mutation, 20 optimisations sont effectuées avec 100 individus sur 200 générations. Même si les différences sont faibles, l'algorithme *Omni-Optimizer* donne statistiquement de meilleurs résultats que l'algorithme *NSGA-II*.

La figure 5.24 montre l'évolution statistique de l'hypervolume relatif pour les deux algorithmes,

avec une probabilité de mutation réelle de 30%. L'algorithme *NSGA-II* présente une convergence initiale plus rapide, toutefois *Omni-Optimizer* parvient à combler son retard et présente à la fin des 200 générations de meilleurs résultats.

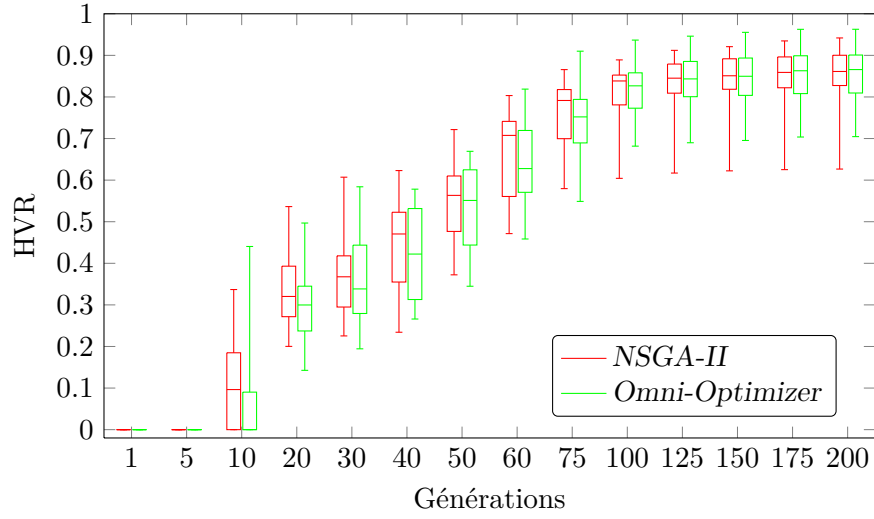


Figure 5.24 – Comparaison de l'évolution des hypervolumes relatifs pour les algorithmes *NSGA-II* et *Omni-Optimizer*. Les statistiques sont réalisées avec 20 optimisations de 100 individus sur 200 générations. La probabilité de mutation p_m est fixée à 30%. Les boîtes à moustaches de la dernière génération se retrouvent sur la figure 5.23.

5.2.7.2 Comparaison sur la distance au front de Pareto

Cette mesure fournit le même genre d'informations que l'hypervolume relatif. La table 5.9 présente les statistiques sur la distance entre les surfaces de compromis et le front de Pareto. Plus cette distance est faible, meilleure est l'approximation du front de Pareto. La définition de cette distance est présentée section 2.4.1.1. Quelle que soit la probabilité de mutation, l'algorithme *Omni-Optimizer* donne de meilleurs résultats.

Algorithme	p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
<i>NSGA-II</i>	10%	0.0439	0.2435	0.4344	0.1184	0.1722	0.2177	0.3434
<i>Omni-Optimizer</i>	10%	0.0165	0.2319	0.4150	0.1080	0.1647	0.2067	0.3248
<i>NSGA-II</i>	20%	0.0834	0.2352	0.4791	0.1187	0.1359	0.2049	0.3246
<i>Omni-Optimizer</i>	20%	0.0041	0.2205	0.4404	0.1048	0.1551	0.2133	0.2639
<i>NSGA-II</i>	30%	0.0485	0.2175	0.4593	0.1031	0.1672	0.2064	0.2535
<i>Omni-Optimizer</i>	30%	0.0040	0.1960	0.3663	0.0768	0.1603	0.1986	0.2253
<i>NSGA-II</i>	40%	0.1405	0.2654	0.4392	0.0858	0.2163	0.2354	0.3218
<i>Omni-Optimizer</i>	40%	0.0102	0.2225	0.4951	0.1069	0.1535	0.2199	0.2772
<i>NSGA-II</i>	50%	0.0702	0.2158	0.4493	0.0902	0.1434	0.2202	0.2786
<i>Omni-Optimizer</i>	50%	0.0282	0.2014	0.4007	0.0942	0.1365	0.2163	0.2458
<i>NSGA-II</i>	60%	0.1045	0.2386	0.4883	0.1002	0.1456	0.2316	0.3049
<i>Omni-Optimizer</i>	60%	0.0820	0.2342	0.5195	0.1033	0.1698	0.2281	0.2813

Table 5.9 – Mesures statistiques de la distance entre la surface de compromis et le front de Pareto relatif pour les algorithmes *NSGA-II* et *Omni-Optimizer*. (Évolution de la probabilité de mutation avec $p_s = 10\%$).

5.2.7.3 Comparaison sur la représentation du front de Pareto

La table 5.10 présente les statistiques sur la représentation du front de Pareto par les surfaces de compromis identifiées obtenues par les algorithmes *NSGA-II* et *Omni-Optimizer* pour différentes

probabilités de mutation. La définition de la représentation du front de Pareto est présentée section 2.4.1.2. Plus les valeurs sont faibles, meilleure est la représentation du front de Pareto. Comme pour la mesure de la distance au front de Pareto, cette mesure donne un léger avantage à *Omni-Optimizer*.

<i>Algorithme</i>	p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
<i>NSGA-II</i>	10%	0.0630	0.2942	0.5428	0.1472	0.1967	0.2568	0.4338
<i>Omni-Optimizer</i>	10%	0.0564	0.2806	0.4971	0.1353	0.1746	0.2541	0.4028
<i>NSGA-II</i>	20%	0.0811	0.2943	0.5957	0.1551	0.1876	0.2550	0.4288
<i>Omni-Optimizer</i>	20%	0.0407	0.2703	0.5748	0.1322	0.1845	0.2308	0.3302
<i>NSGA-II</i>	30%	0.0775	0.2546	0.5195	0.1187	0.1673	0.2601	0.3031
<i>Omni-Optimizer</i>	30%	0.0402	0.2270	0.4454	0.0811	0.1888	0.2222	0.2642
<i>NSGA-II</i>	40%	0.1268	0.3044	0.5647	0.1131	0.2373	0.2632	0.3677
<i>Omni-Optimizer</i>	40%	0.0388	0.2554	0.5223	0.1221	0.1653	0.2549	0.3323
<i>NSGA-II</i>	50%	0.0809	0.2436	0.4923	0.0994	0.1752	0.2294	0.3208
<i>Omni-Optimizer</i>	50%	0.0655	0.2273	0.4720	0.1050	0.1367	0.2446	0.2771
<i>NSGA-II</i>	60%	0.0841	0.2624	0.5822	0.1198	0.1690	0.2472	0.3452
<i>Omni-Optimizer</i>	60%	0.0946	0.2651	0.5908	0.1280	0.1559	0.2790	0.3478

Table 5.10 – Mesures statistiques de la représentation du front de Pareto par les surfaces de compromis obtenues par les algorithmes *NSGA-II* et *Omni-Optimizer* pour différentes probabilités de mutation. ($p_s = 10\%$). Plus les valeurs sont faibles, meilleure est la représentation.

5.2.7.4 Comparaison sur la diversité phénotypique

Les algorithmes *NSGA-II* et *Omni-Optimizer* utilisent tous les deux un mécanisme de diversité phénotypique. Ce mécanisme a pour objectif de maintenir une certaine diversité sur la surface de compromis. Les données statistiques de cette diversité sont présentées table 5.11. Plus les valeurs sont faibles, meilleure est la distribution des solutions le long de la surface de compromis. Les résultats donnent ici l'avantage à *NSGA-II*. Contrairement à *Omni-Optimizer*, *NSGA-II* prend uniquement en compte les distances phénotypiques, c'est donc normal que sur ce type de mesure il obtienne de meilleurs résultats que *Omni-Optimizer*.

<i>Algorithme</i>	p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
<i>NSGA-II</i>	10%	0.3033	0.5860	0.8635	0.1725	0.4388	0.6069	0.7283
<i>Omni-Optimizer</i>	10%	0.2524	0.6047	1.0846	0.2019	0.4846	0.5980	0.6901
<i>NSGA-II</i>	20%	0.2497	0.6061	0.9790	0.2274	0.3914	0.6213	0.8319
<i>Omni-Optimizer</i>	20%	0.2557	0.6136	1.0457	0.2270	0.4288	0.5836	0.7935
<i>NSGA-II</i>	30%	0.1997	0.4934	0.8933	0.1999	0.3704	0.4498	0.6113
<i>Omni-Optimizer</i>	30%	0.2709	0.5253	0.9699	0.2086	0.3428	0.4775	0.6184
<i>NSGA-II</i>	40%	0.2299	0.5152	0.9010	0.1890	0.3343	0.4822	0.6419
<i>Omni-Optimizer</i>	40%	0.2926	0.5517	0.8880	0.1500	0.4216	0.5499	0.6372
<i>NSGA-II</i>	50%	0.2058	0.5663	1.1389	0.2497	0.4543	0.5074	0.7086
<i>Omni-Optimizer</i>	50%	0.3198	0.5574	1.0539	0.2086	0.3737	0.5103	0.6021
<i>NSGA-II</i>	60%	0.2152	0.5309	0.8983	0.1988	0.3748	0.4561	0.6625
<i>Omni-Optimizer</i>	60%	0.2863	0.5391	0.9547	0.1751	0.4150	0.5118	0.6442

Table 5.11 – Mesures statistiques de la diversité phénotypique pour les algorithmes *NSGA-II* et *Omni-Optimizer*. (Évolution de la probabilité de mutation avec $p_s = 10\%$).

5.2.7.5 Comparaison sur la diversité génotypique

L'algorithme *Omni-Optimizer* utilise un mécanisme de préservation de diversité génotypique. Ce mécanisme est censé assurer le maintien de la diversité sur les variables, l'objectif étant d'avoir la

diversité la plus grande. Pour évaluer cette diversité, on construit une distance propre au problème de placement. Cette distance génotypique mesure les différences entre deux solutions. Elle est définie de la manière suivante :

$$\mathcal{D}(\mathbf{s}_i, \mathbf{s}_j) = \sqrt{\frac{1}{2n_{\text{real}}} \sum_{k=1}^{n_{\text{real}}} \left(\frac{x_{i,k} - x_{j,k}}{u_k^u - u_k^l} \right)^2 + \frac{1}{2n_{\text{disc}}} \sum_{k=1}^{n_{\text{disc}}} \left(\frac{\min\{|\theta_{i,k} - \theta_{j,k}|, 2\pi - |\theta_{i,k} - \theta_{j,k}|\}}{\pi} \right)^2} \quad (5.3)$$

où \mathbf{s}_i et \mathbf{s}_j représentent les données des solutions i et j . La première partie de cette distance fait intervenir les positions des composants, normées par rapport aux bornes des variables (u_k^u et u_k^l sont les bornes inférieures et supérieures de la variable k). La seconde partie contient une distance entre les angles de positionnement des composants. Cette dernière prend en compte la périodicité des angles. La distance \mathcal{D} est normée de telle manière qu'elle soit comprise entre zéro et un.

Pour chaque optimisation, les distances génotypiques entre les solutions des surfaces de compromis peuvent être évaluées. Ce calcul à complexité quadratique peut être résumé dans une matrice triangulaire supérieure \mathbf{M} , où M_{ij} est la distance génotypique entre les solutions i et j ($j > i$). Le nombre de lignes et de colonnes de cette matrice correspond au nombre de solutions efficaces trouvées lors de l'optimisation. Ce nombre est noté n_{eff} . La comparaison entre de tels résultats peut s'avérer difficile entre deux optimisations différentes. Pour cette raison, les valeurs moyenne et maximale de ces matrices seront donc utilisées pour établir des comparaisons et des statistiques.

$$\begin{aligned} \mathcal{D}_{\text{moy}} &= \frac{2}{n_{\text{eff}} \times (n_{\text{eff}} - 1)} \sum_{i=1}^{n_{\text{eff}}-1} \sum_{j=i+1}^{n_{\text{eff}}} \mathcal{D}(\mathbf{s}_i, \mathbf{s}_j) \\ \mathcal{D}_{\text{max}} &= \max_{1 \leq i < j \leq n_{\text{eff}}} \{\mathcal{D}(\mathbf{s}_i, \mathbf{s}_j)\} \end{aligned} \quad (5.4)$$

La figure 5.25 présente graphiquement les statistiques, effectuées sur 20 optimisations différentes avec 100 individus sur 200 générations et ce pour 6 variations de la probabilité de mutation des variables réelles. Les statistiques réalisées sur les 240 optimisations montrent que les mécanismes de préservation de diversité génotypique de l'algorithme *Omni-Optimizer* remplissent leur mission. Globalement, les moyennes et médianes calculées sur \mathcal{D}_{moy} et \mathcal{D}_{max} sont plus élevées. Les tables C.10 et C.11, présentées en annexe page 167, résument numériquement les statistiques sur les distances génotypiques moyennes et maximales.

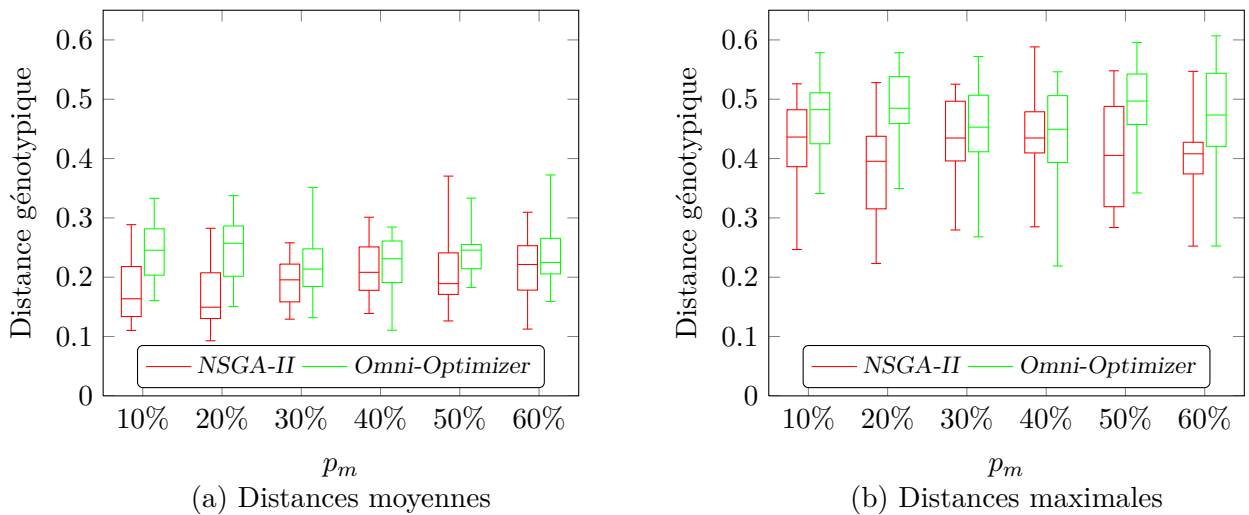


Figure 5.25 – Comparaison des distances génotypiques moyennes et maximales pour les algorithmes *NSGA-II* et *Omni-Optimizer*.

5.2.8 Rôle de la population initiale

Les résultats précédents ont montré que la méthode proposée était capable d'identifier des solutions réalisables à partir de populations initiales aléatoires. Toutefois, les solutions efficaces issues de chaque

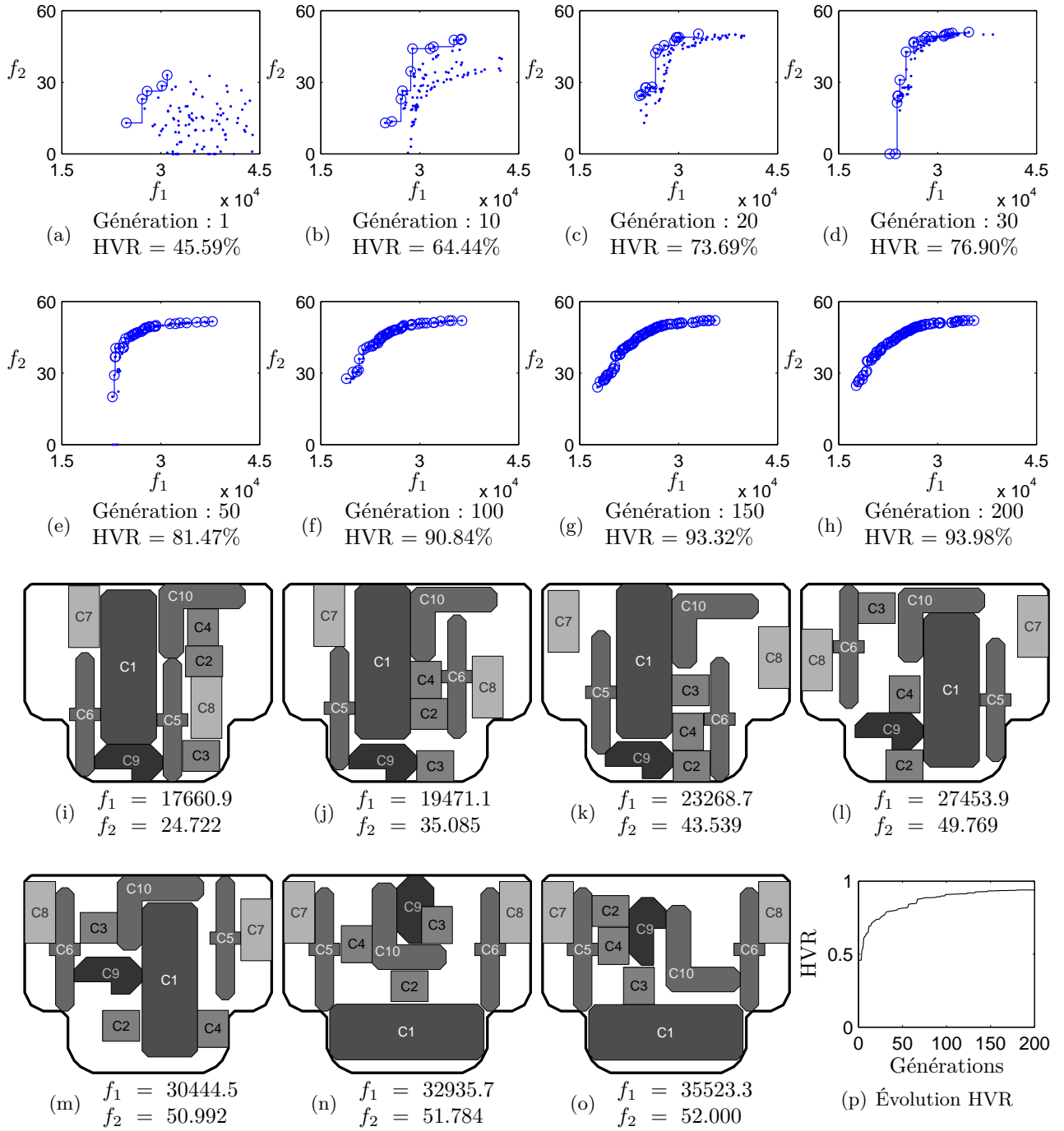


Figure 5.26 – Évolution de la convergence et présentation des solutions efficaces pour une optimisation réalisée avec des solutions initiales réalisables. Ces dernières ont été générées à l'aide de l'heuristique de placement *Bottom-Left-Fill* présentée page 67. Chaque point représente une solution réalisable. Les points non dominés sont cerclés. La sous-figure (p) montre l'évolution de l'hypervolume relatif au cours des générations.

optimisation sont assez proches les unes des autres dans l'espace des variables.

Ces différents éléments nous ont poussé à analyser le rôle joué par la population initiale. Pour cela, l'heuristique de placement de type *Bottom-Left-Fill* présentée section 4.2.5 a été utilisée pour générer des solutions initiales réalisables. Les directions de dépôt des composants ont été alternées pour obtenir des solutions différentes.

Les premiers résultats montrent qu'une optimisation est capable de proposer des solutions très différentes. Les surfaces de compromis présentent une distance phénotypique maximale plus importante et davantage de discontinuités, signe de la présence de différentes configurations topologiques dans les solutions. Auparavant, les surfaces de compromis étaient continues, car l'ensemble des solutions efficaces proposées étaient issues d'une des premières solutions réalisables identifiées. La figure 5.26 présente la convergence du problème pour une population initiale dont l'ensemble des solutions est réalisable. Dès la première génération, la valeur de l'hypervolume relatif approche les 50%, et la surface de compromis comporte déjà cinq solutions. Le fait d'avoir plusieurs solutions efficaces initialement permet de faire évoluer plusieurs solutions sur la surfaces de compromis et ainsi de répartir de manière plus uniforme la pression de sélection.

5.2.9 Discussion

Cet exemple a servi à justifier les choix algorithmiques réalisés. L'algorithme proposé est capable, à partir de solutions aléatoires, de trouver des solutions réalisables. Les différentes optimisations réalisées ont permis de mettre en avant le rôle de l'algorithme de séparation, le rôle des opérateurs génétiques utilisés. Le réglage des paramètres de l'algorithme génétique a un rôle important dans la convergence du problème. L'analyse des résultats de cet exemple permet de proposer des recommandations quant à l'utilisation des opérateurs génétiques. Classiquement, les opérateurs de croisement sont réglés avec une faible probabilité d'action (moins de 10%). L'opérateur de mutation joue un rôle important dans l'évolution de la convergence, et l'opérateur d'échange permet d'accélérer la convergence initiale du problème.

La comparaison de deux optimisations avec deux populations initiales différentes a montré que les surfaces de compromis ne sont pas très différentes, mais que les solutions proposées ont généralement peu de choses en commun.

Même si l'algorithme *Omni-Optimizer* a montré numériquement qu'il présentait des résultats sensiblement meilleurs que l'algorithme *NSGA-II* tant au niveau de la convergence que de la diversité génotypique, il faut réfléchir à un autre mécanisme pour préserver davantage de diversité génotypique. L'utilisation d'une population initiale réalisable est un moyen simple pour maintenir la diversité génotypique.

5.3 Résultats obtenus pour un 2^{ème} problème d'agencement

Cet exemple illustre les capacités de l'optimiseur pour un problème présentant une forte compacité.

5.3.1 Données du problème

Dans ce troisième exemple, on considère toujours un problème d'optimisation bi-objectif avec les composants du problème 2, auxquels on ajoute un composant de type *T1* supplémentaire. La compacité passe alors de 59.7% à 79.5%. La figure 5.27 présente l'ensemble de ces composants. Avec 11 composants, le nombre de variables réelles passent à 22 et 11 variables discrètes codent à présent les orientations des composants.

Deux objectifs sont toujours pris en compte : le premier consiste à minimiser la distance euclidienne entre les centres de gravité du contenant et de l'assemblage des composants. Le second objectif consiste à minimiser le moment d'inertie de l'assemblage autour de l'axe *y* du coffre (Figure 5.2(a)). Le premier objectif caractérise un comportement statique, alors que le deuxième renseigne sur un comportement dynamique.

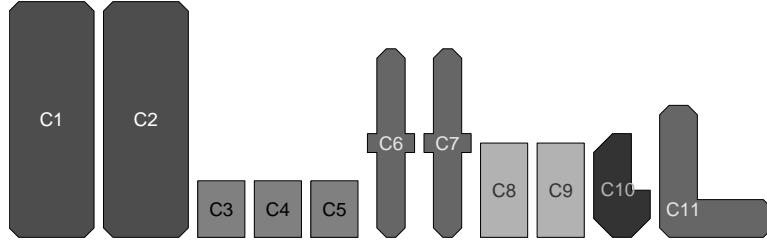


Figure 5.27 – Présentation des différents composants à positionner.

$$\mathcal{P} \begin{cases} \min f_1(\mathbf{v}) = \sqrt{d_x^2 + d_y^2} \\ \min f_2(\mathbf{v}) = \sum_{i=1}^m \rho_i I_{i/yy} \\ \text{t.q. les contraintes de placement soient satisfaites} \end{cases} \quad \text{où} \quad \begin{cases} d_x = \frac{1}{\sum_{i=1}^m \rho_i A_i} \sum_{i=1}^m \rho_i A_i (x_i - x_G) \\ d_y = \frac{1}{\sum_{i=1}^m \rho_i A_i} \sum_{i=1}^m \rho_i A_i (y_i - y_G) \end{cases}$$

5.3.2 Premières optimisations

La figure 5.28 présente les différentes surfaces de compromis obtenues pour différentes populations initiales aléatoires (12 au total). Les points matérialisés avec les mêmes symboles représentent l'ensemble des points non dominés obtenus pour une même optimisation. L'analyse de cette figure révèle de grandes disparités et montre que les résultats obtenus dépendent de la population initiale. Ceci peut s'expliquer par plusieurs facteurs : l'analyse de la convergence de l'algorithme génétique révèle qu'il est difficile de trouver des solutions admissibles pour un problème si compact. Il faut attendre de plus en plus de générations avant de voir apparaître des solutions réalisables. Les contraintes de placement deviennent de plus en plus difficiles à satisfaire avec l'augmentation de la compacité. Une fois qu'une solution admissible est trouvée, cette solution exerce une pression de sélection sur les autres solutions, et la population a tendance à perdre sa diversité initiale. Par conséquent, les solutions issues d'une même surface de compromis présentent une grande similarité, même si la modélisation essaie de préserver les diversités phénotypiques et génotypiques.

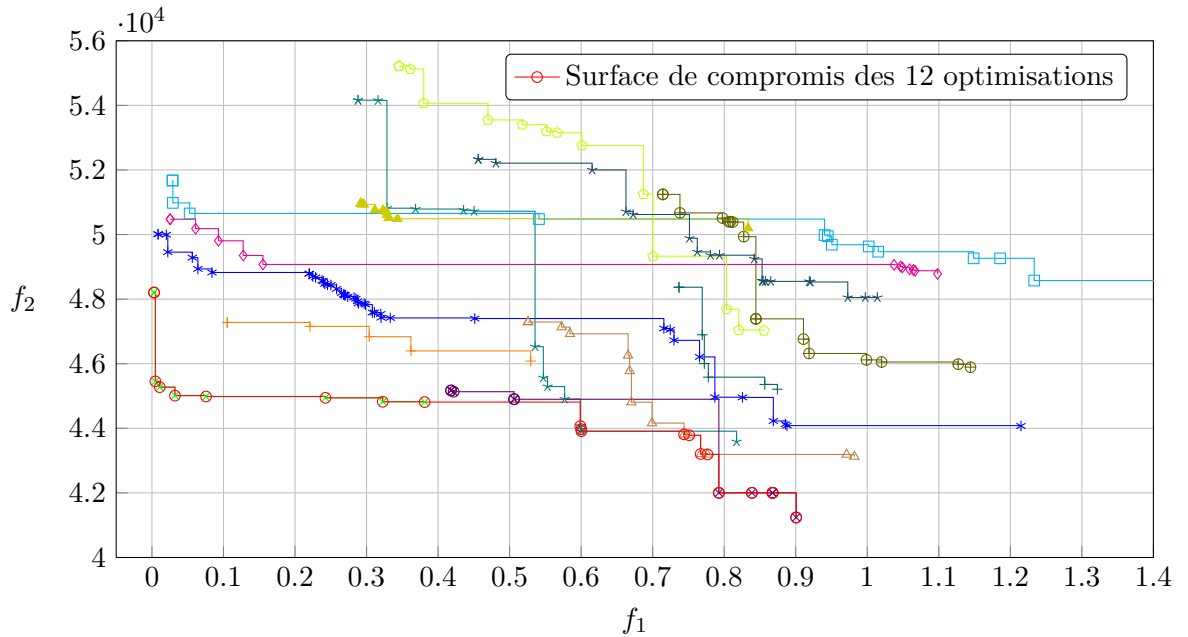


Figure 5.28 – Surfaces de compromis des douze premières optimisations réalisées. La surface de compromis calculée à partir des meilleures solutions des douze optimisations est aussi représentée.

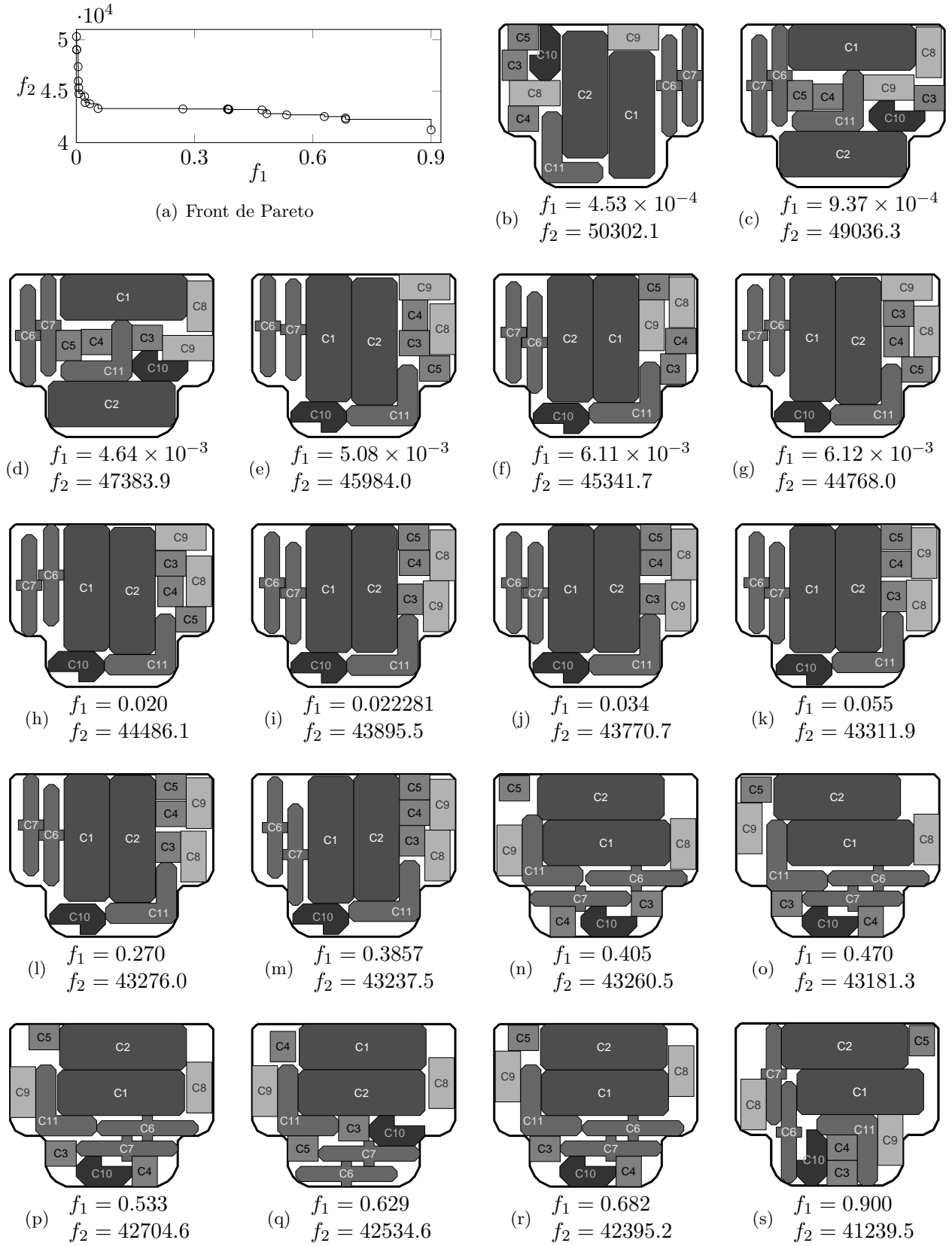


Figure 5.29 – Front de Pareto pour le second problème d’agencement avec la présentation de quelques solutions efficaces.

5.3.3 Solutions du front de Pareto

Les différentes optimisations réalisées sont utilisées pour évaluer la meilleure approximation du front de Pareto. La figure 5.29 présente le front de Pareto identifié ainsi que quelques-unes des solutions efficaces du problème. Ces solutions sont triées selon l'ordre croissant de l'objectif 1 et décroissant pour l'objectif 2. Sur chacune des sous-figures, on peut observer que la majorité des composants est en contact, et qu'il existe peu de jeu pour déplacer les composants. Même pour un problème à forte compacité, on peut observer la diversité des solutions efficaces qui existe. La figure 5.30 présente un ensemble de solutions sous-optimales. Là encore, il existe des solutions proches du front de Pareto mais très différentes de celles présentes sur le front de Pareto. Le point de référence utilisé pour évaluer l'hypervolume a pour coordonnées $(1.5, 5.7 \times 10^4)$. L'hypervolume de référence est évalué à 2.1453×10^4 .

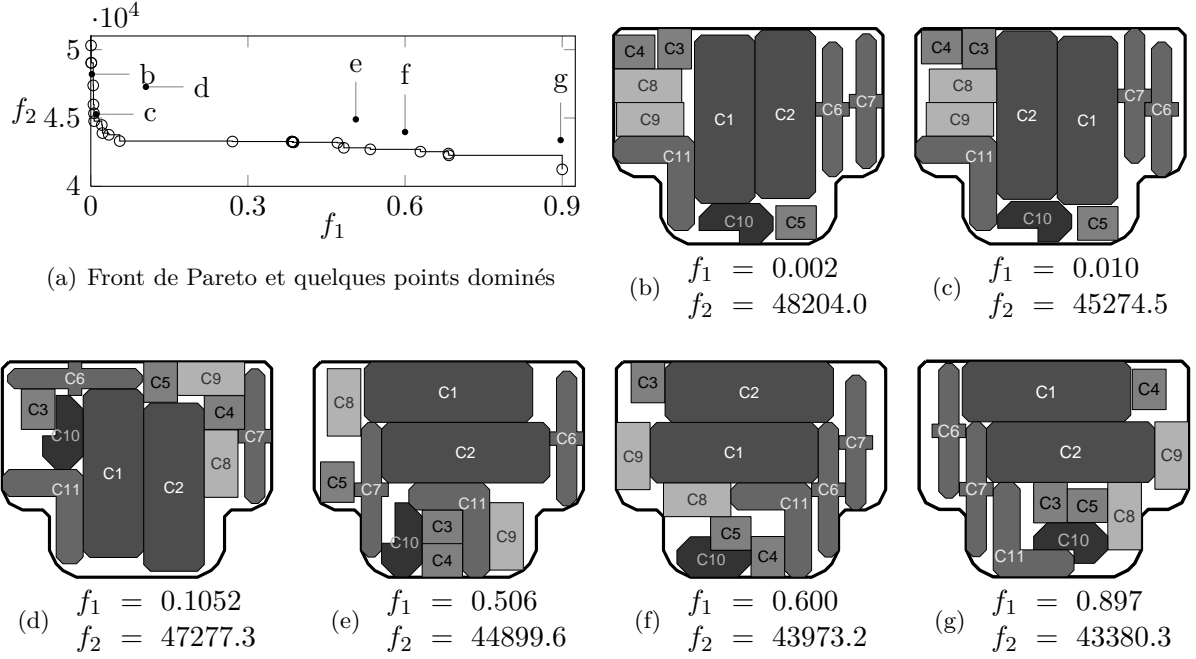


Figure 5.30 – Présentation de quelques-unes des solutions sous-optimales pour le problème de placement.

5.3.4 Réglages des opérateurs génétiques

Les premières optimisations réalisées montrent comme il est difficile pour l'optimiseur de trouver des solutions efficaces identiques d'une optimisation à une autre. Pour obtenir de meilleurs résultats, on peut étudier la sensibilité des paramètres de l'algorithme d'optimisation.

Les résultats présentés ici montrent l'influence des paramètres des probabilités de mutation et d'échange sur la convergence des problèmes. La figure 5.31 présente les différentes surfaces de compromis obtenues pour des variations p_m allant de 0% à 100% avec $p_s = 30\%$. Pour chaque variation du paramètre p_m , on effectue 50 optimisations et on en extrait la meilleure surface de compromis. D'une variation du paramètre p_m à l'autre, les populations initiales sont identiques. Les autres paramètres de l'algorithme sont donnés dans le tableau 5.4. Chaque point de cette figure représente une solution réalisable. Même en comparant les meilleures solutions de 50 optimisations, les surfaces de compromis sont différentes, cela montre bien l'importance de réglage des paramètres génétiques. On observe sur cette figure une amélioration des résultats lors de la progression du paramètre de probabilité p_m allant de 0% jusqu'à 20%, puis une dégradation de ces mêmes résultats pour des valeurs supérieures. Le problème apparaît donc très sensible aux réglages du paramètre de mutation p_m . La figure 5.32 présente les différentes surfaces de compromis obtenues pour des variations p_s allant de 0% à 100% avec $p_m = 30\%$. Les différences sont moins visibles par rapport aux variations de l'opérateur de mutation, ce qui laisse à penser que l'opérateur d'échange a un rôle moins important. La figure 5.33 présente

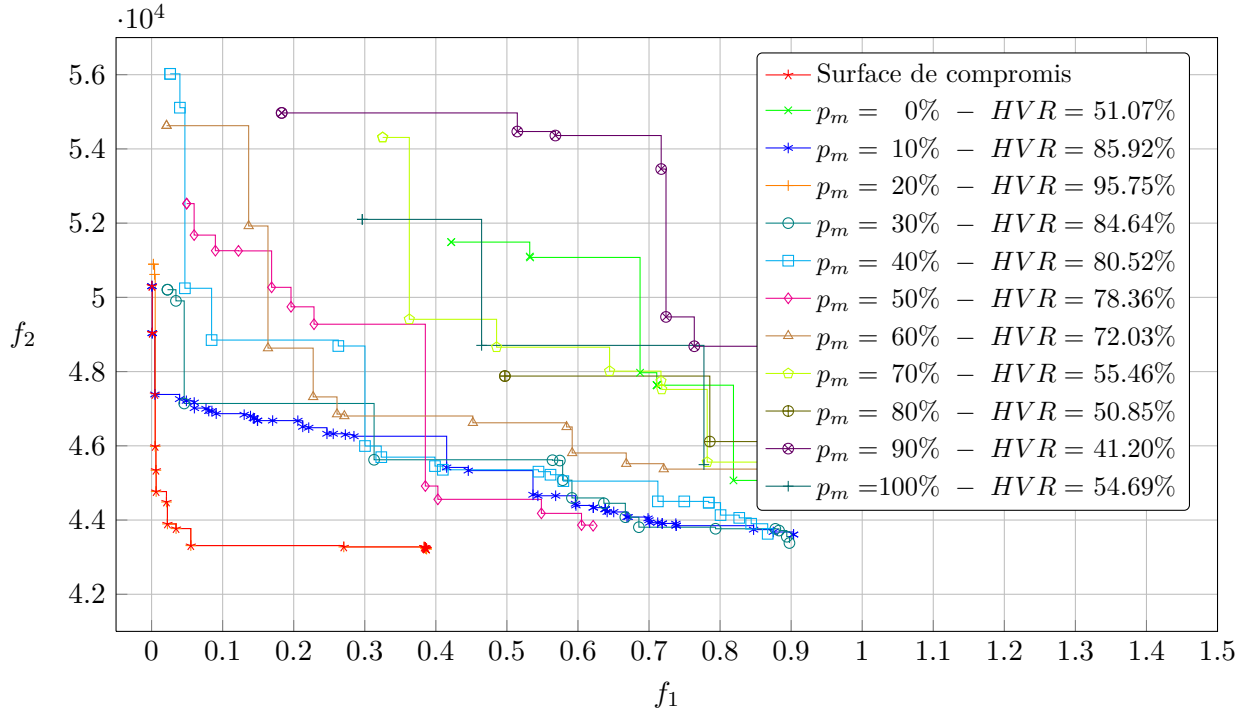


Figure 5.31 – Influence de la probabilité de mutation pour le problème d’agencement 2. La probabilité de l’opérateur d’échange est $p_s = 30\%$. La surface de compromis de toutes les optimisations réalisées ici représente 95.85% de l’hypervolume total.

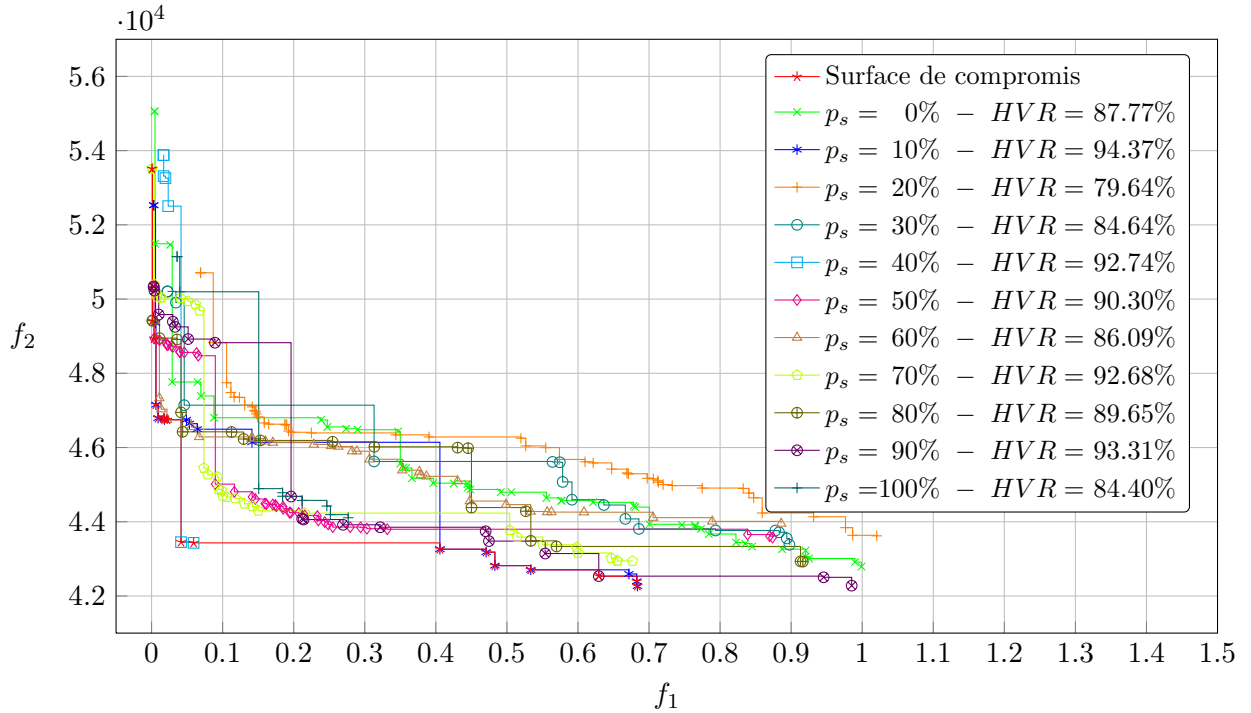


Figure 5.32 – Influence de la probabilité d’échange pour le problème d’agencement 2. La probabilité de l’opérateur de mutation est $p_m = 30\%$. La surface de compromis de toutes les optimisations réalisées ici représente 99.31% de l’hypervolume total.

les statistiques de l'hypervolume relatif pour les différentes optimisations réalisées en fonction des probabilités de mutation p_m et d'échange p_s . Les données statistiques de cette figure sont regroupées en annexe dans les tables C.12 et C.13. Le rôle de l'opérateur de mutation apparaît visiblement sur la sous-figure 5.33(a). En revanche, l'influence de l'opérateur d'échange apparaît moins visiblement sur la sous-figure 5.33(b). Sur ces deux sous-figures, les quatrièmes boîtes à moustaches correspondant à des probabilités de 30% sont identiques. En effet, elles correspondent aux mêmes ensembles d'optimisations où $p_m = p_s = 30\%$. Cette observation se retrouve sur les figures 5.31 et 5.32 où les surfaces de compromis et les valeurs d'hypervolume relatif sont identiques pour les probabilités de 30%. Les valeurs minimales de l'ensemble de ces statistiques sont nulles, indiquant que pour chaque série d'optimisations au moins une optimisation n'a pas réussi à identifier une solution réalisable. Si on regarde les valeurs de l'indice de violation de contraintes de ces simulations, les solutions finales présentent un indice proche de zéro (10^{-2} à 0.2). Ces solutions présentent donc de très légers chevauchements. L'utilisation de solutions initiales réalisables aurait permis de proposer des solutions réalisables. Une autre idée consiste à relaxer les contraintes de placement.

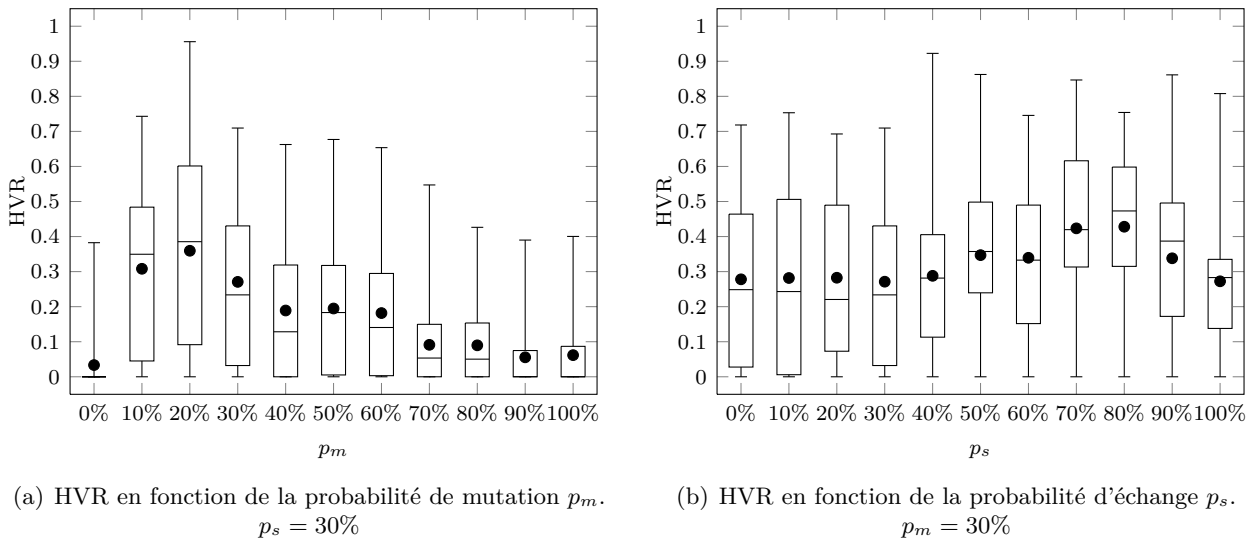


Figure 5.33 – Comparaison statistique de l'évolution des valeurs de l'hypervolume relatif pour des variations sur les probabilités de mutation p_m et d'échange p_s . Chaque boîte à moustaches a été calculée avec 50 optimisations. Les 4^{ème} boîtes à moustaches obtenues pour $p_m = p_s = 30\%$ sont identiques sur les deux sous-figures (Optimisations identiques).

5.3.5 Relaxation des contraintes de placement

Les résultats précédents montrent qu'il est difficile de trouver des solutions initiales réalisables. On a vu que l'initialisation des solutions à l'aide d'une heuristique de placement permettait de surmonter cette difficulté. On se propose ici de tester une relaxation des contraintes de placement.

Actuellement, une solution est dite réalisable si la valeur renvoyée par l'algorithme de séparation est nulle. Pour faciliter la convergence du problème et maintenir la diversité des solutions, on peut considérer qu'une solution est réalisable si la valeur de l'indice de violation de contraintes est inférieure à une valeur ε fixée par le concepteur. Les solutions ainsi produites présenteront ainsi quelques légers chevauchements. Cette piste est intéressante pour les problèmes dont la géométrie des composants n'est définitive et peuvent subir quelques variations géométriques. Cela revient à généraliser la notion d' ε -dominance, utilisée pour relaxer le critère de dominance dans l'espace des objectifs [DT08].

La figure 5.34 présente le front de Pareto et un sous-ensemble des solutions efficaces obtenus pour ce problème avec une relaxation des contraintes de placement. Le front de Pareto a été obtenu à l'aide 50 optimisations. Les solutions sont considérées réalisables dès lors que la valeur de la fonction de séparation est inférieure 0.5. Pour information, les solutions d'une population initiale présentent en moyenne pour cette fonction une valeur de 50. La sous-figure (a) montre l'influence de la relaxation des contraintes sur le front de Pareto. Cette relaxation permet d'améliorer grandement le front de

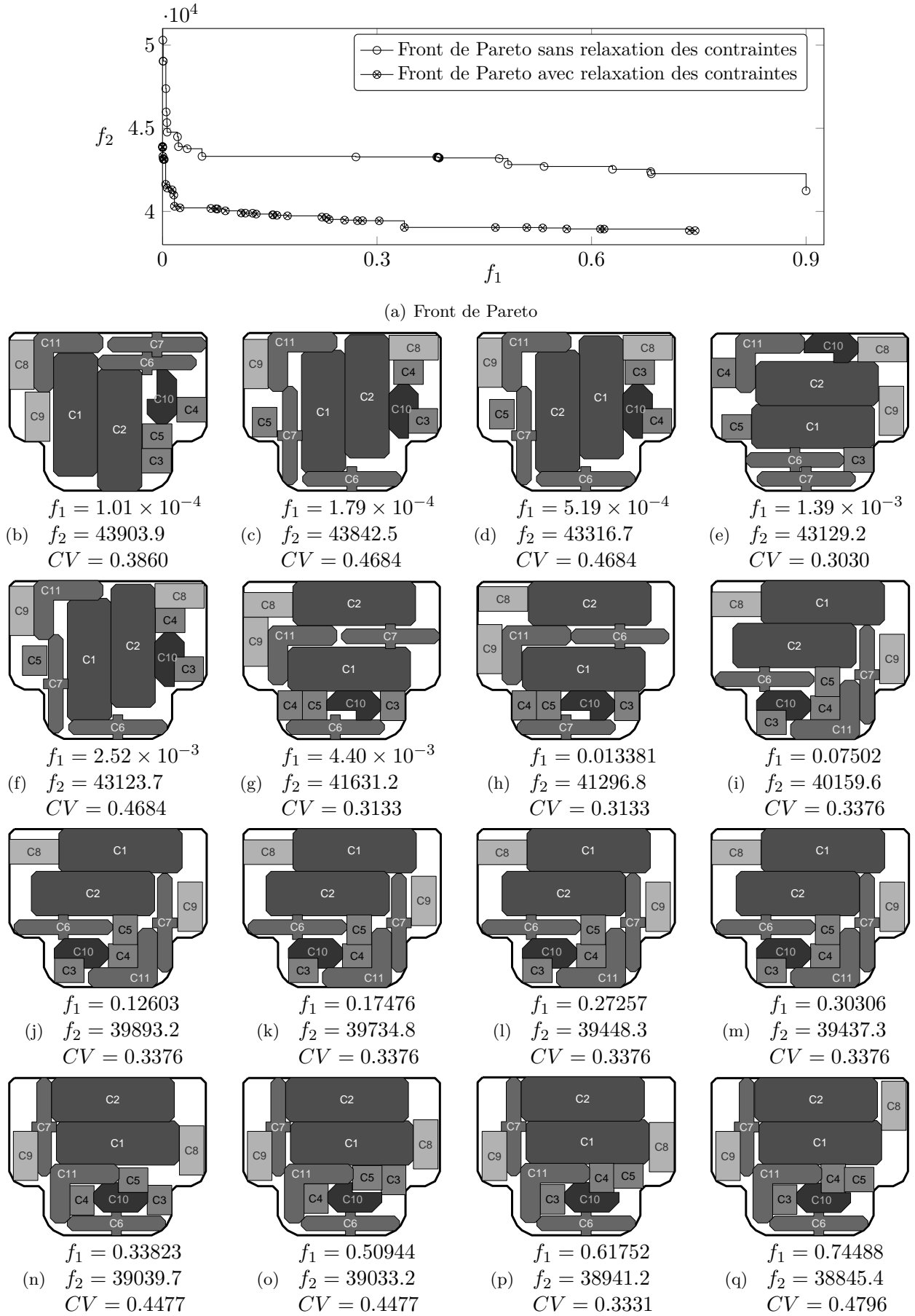


Figure 5.34 – Solutions extraites du front de Pareto avec une relaxation de 0.5 sur les contraintes de placement.

Pareto. Les sous-figures (b) à (q) présentent les solutions efficaces avec la relaxation. Pour chacune d'entre elles, les valeurs des objectifs f_1 et f_2 sont données, et l'indice de violation de contrainte est mentionné. Tous ces indices ont une valeur inférieure au seuil de 0.5 fixé. On peut constater que ces solutions n'ont pas grand chose à voir avec les solutions non relaxées présentées figure 5.29, autrement dit que les distances génotypiques entre ces deux ensembles de solutions sont importantes. On peut aussi noter qu'il est difficile de modifier manuellement ces solutions pour obtenir des solutions ne présentant aucun chevauchement.

Si la géométrie des composants n'est pas définitive, une fine relaxation des contraintes de placement permet d'améliorer les résultats tant au niveau du front de Pareto que de la diversité des solutions pouvant exister. La relaxation présentée ici était valable pour tous les composants, on peut personnaliser la relaxation des contraintes : pour chaque composant, on peut définir un seuil de tolérance pour la contrainte d'appartenance et aussi une matrice de seuils de tolérance pour les contraintes de non-chevauchement entre chaque paire de composants.

5.4 Conclusion

Les résultats présentés ont permis de tester la méthode de placement proposée dans le chapitre 4. Ces résultats ont montré que notre méthode est capable de résoudre des problèmes de découpe comme des problèmes d'agencement. Sur le premier exemple présenté, notre méthode a été comparée avec les résultats de référence issus de la littérature sur le problème de découpe de formes irrégulières. On a pu constater qu'on obtenait des résultats proches de ceux obtenus avec des méthodes dédiées en un temps de calcul très raisonnable (20 minutes).

Concernant les problèmes d'agencement que nous avons créés, la méthode est capable de trouver des solutions réalisables à partir de populations initiales aléatoires. Elle permet de résoudre des problèmes d'agencement avec une compacité supérieure à 80%. Dans ce cas là, le problème devient quasiment un problème C&P, pour lequel trouver une solution réalisable devient un vrai défi.

Les différentes optimisations réalisées ont permis de mettre en avant le rôle des différents opérateurs. L'utilisation de l'algorithme de séparation est indispensable pour obtenir des solutions réalisables. Les opérateurs de mutation et d'échange sont apparus comme les moteurs de convergence du problème. Les différents indicateurs multi-objectifs ont permis de comprendre les étapes de convergence de la méthode. La comparaison des algorithmes *Omni-Optimizer* et *NSGA-II* a montré que *Omni-Optimizer* permettait d'obtenir une meilleure convergence ainsi qu'une meilleure diversité génotypique. Enfin, les différentes optimisations réalisées ont permis de préconiser des réglages pour l'algorithme génétique.

La faible diversité génotypique observée avec les initialisations aléatoires des populations de l'algorithme génétique a été palliée par l'utilisation de solutions initiales réalisables. Pour maintenir la diversité génotypique, d'autres pistes doivent être explorées, notamment sur la manière dont est prise en compte la distance génotypique, ou encore en utilisant des populations d'ensemble de solutions comme Grignon *et al.* l'ont déjà fait [GF99]. Enfin, d'autres opérations de croisement spécifiques aux problèmes de placement pourraient être envisagées.

Conclusions et perspectives

Dans ce travail, nous nous sommes intéressés à l'optimisation d'agencements géométriques et fonctionnels. Nos principaux buts étaient de lister et catégoriser l'ensemble de ces problèmes, d'analyser les méthodes de résolution existantes et proposer un cadre de travail ouvert capable de traiter des problèmes génériques avec des objets de géométries complexes et permettant l'intégration de toute sorte de contraintes et d'objectifs.

Après avoir réalisé un état de l'art des différents problèmes de placement, regroupant les problèmes de découpe, de conditionnement et d'agencement, nous avons résumé l'ensemble des techniques de résolution et d'optimisation, qui ont été développées. Ce premier chapitre a permis de mettre en évidence les difficultés de formulation et de résolution de ces problèmes, ainsi que les caractères spécifiques et mono-objectif des méthodes de résolution existantes.

Le second chapitre a présenté la théorie de l'optimisation multi-objectif ainsi que les différentes techniques de résolution existantes. Les différents algorithmes d'optimisation évolutionnaires utilisés dans la méthode de résolution proposée sont présentés dans ce chapitre. Enfin, les outils d'analyse de convergence multi-objectif, qui ont été utilisés dans les différents exemples de ce mémoire, ont été décrits.

Le troisième chapitre est un exemple de résolution multi-objectif d'un problème d'agencement de compartiments de navire. Cet exemple a été formulé comme un problème d'optimisation combinatoire bi-objectif, où l'on cherche à minimiser les flux entre les compartiments du navire et à maximiser une fonction d'adjacence définie par le concepteur. La modélisation proposée ne se limite pas à l'agencement des navires, mais s'applique à tous les problèmes d'agencement pouvant être modélisé comme un placement séquentiel, où les composants / compartiments sont positionnés les uns après les autres. Toutefois, cette modélisation ne peut pas être appliquée à tous les problèmes de placement, ce qui nous a amenés à développer une méthode de résolution générale.

Dans le chapitre 4, nous avons proposé une méthode générique pour la résolution des problèmes de placement multi-objectif faisant intervenir la géométrie des composants. Nous avons choisi de développer une méthode de placement relaxé pour sa souplesse d'adaptation aux différents problèmes rencontrés. La méthode proposée est une hybridation d'un algorithme évolutionnaire avec un algorithme de séparation. L'algorithme évolutionnaire choisi est un algorithme génétique multi-objectif chargé d'explorer efficacement l'espace de recherche. L'algorithme de séparation a pour objectif de faire respecter les contraintes de placement du problème. Concrètement, avant d'évaluer les objectifs de chaque solution, les contraintes de placement sont vérifiées. Si celles-ci ne sont pas satisfaites, l'algorithme de séparation modifie la solution de manière à la rendre réalisable. Il effectue pour cela la minimisation d'une fonction continue caractérisant le non-respect des contraintes de placement. L'algorithme de séparation permet de traiter des cas simples comme des cas compliqués, en 2D et 3D, et permet la gestion de contraintes particulières. Pour les problèmes impliquant des rectangles ou des parallélogrammes, une version spécifique de l'algorithme de séparation a été développée permettant ainsi leur résolution de manière efficace. Dans le cas où les composants sont des polygones à orientations discrètes, une méthode basée sur les polygones de non-recouvrement et d'appartenance a été utilisée pour établir les directions de séparation de manière simple et efficace. Dans tous les autres cas, les composants sont approximés à l'aide d'assemblages de cercles ou de sphères pour permettre une détection rapide des collisions. Nous avons montré que ces décompositions doivent être basées sur les

axes médians des composants afin de maximiser l'efficacité des algorithmes de séparation. Nous avons aussi proposé une version de l'algorithme adaptée aux contenants de géométrie complexe, qui permet désormais de traiter toute sorte de forme de contenant.

Le dernier chapitre contient différents exemples 2D avec plusieurs analyses permettant de mettre en évidence les mécanismes mis en jeu pour la résolution des problèmes de placement. Les différents exemples traités montrent que la méthode peut résoudre aussi bien des problèmes de découpe que des problèmes d'agencement avec de fortes compacités ou des contraintes complexes. Le premier exemple a montré que la méthode peut traiter des problèmes de découpe, et s'approcher des résultats de référence de la littérature obtenus avec des méthodes dédiées. Les deux exemples suivants ont permis d'analyser la méthode proposée. Les résultats montrent que l'algorithme de séparation joue un rôle essentiel lors des résolutions. Le couplage des algorithmes d'optimisation globale et locale permet de trouver des solutions réalisables à partir de solutions initiales aléatoires. Pour les problèmes le permettant, des heuristiques de placement peuvent être utilisées pour générer des solutions initiales réalisables. Cela facilite la convergence du problème et permet un maintien de la diversité génotypique des solutions. Nous avons aussi montré que la relaxation des contraintes de placement facilite la convergence du problème et améliore les surfaces de compromis. Ce type de relaxation peut s'avérer très utile lorsque la géométrie des composants est autorisée à subir quelques variations. Enfin, les différentes simulations réalisées permettent de proposer des recommandations quant aux réglages des algorithmes génétiques.

Les différents travaux effectués dans cette thèse *CIFRE* ont pour objectif final la mise en place d'un outil industriel pour la résolution des problèmes d'agencement. Dans cette optique, une réflexion sur la standardisation de formulation de ces problèmes a été élaborée en vue de leur intégration dans un outil de résolution. À partir de cette réflexion, le développement d'un démonstrateur 3D a été initié. Ce démonstrateur a pour objectif de tester les différentes méthodes de résolution proposées et de permettre l'interaction entre le concepteur et les solutions proposées. L'annexe D présente les premiers travaux réalisés.

L'approche générique proposée comporte de nombreux avantages et possibilités qui n'ont pas tous été exploités, qui pourront donner lieu à des développements futurs. Ces développements incluent :

- l'utilisation d'un algorithme à génération continue. L'algorithme génétique utilisé actuellement est générationnel. Cela signifie que les solutions qui viennent d'être évaluées ne peuvent pas être utilisées avant la génération suivante. L'utilisation d'un algorithme à génération continue permet d'exploiter ces solutions fraîchement calculées. Cela devrait permettre d'accélérer la convergence de résolution, notamment lorsque les solutions ne sont pas réalisables. Le surcoût des opérations génétiques liées au passage à un algorithme à génération continue devrait être négligeable devant le coût des optimisations continues réalisées pour la séparation ;
- l'utilisation d'opérateurs génétiques intelligents. Actuellement, les modifications apportées à une solution sont gérées par des opérateurs génétiques standards. Ces derniers appliquent des changements dans les solutions sans savoir s'ils vont les améliorer. Cagan *et al.* [ACS07a] ont mis en place une sélection des modifications susceptibles d'avoir le maximum d'influence sur les fonctions objectifs. On pourrait envisager de développer une telle approche, de manière à proposer des modifications de solutions exploitant au maximum la connaissance du problème que l'on peut avoir ;
- l'utilisation d'un placement relatif pour modéliser de nouvelles contraintes de placement. Dans les problèmes d'agencement, le placement des composants répond à des contraintes fonctionnelles. L'utilisation d'un repère relatif permet de gérer certaines de ces contraintes et présente de nombreux avantages. Il permet de satisfaire les contraintes fonctionnelles (comme l'alignement de composants, ou la mise à distance). Il permet enfin de réduire l'espace de recherche et par conséquent facilite la convergence du problème. L'intégration de ces contraintes ne pose pas de

difficultés théoriques, elle nécessite seulement des développements techniques.

- la prise en compte de l’agencement dynamique. Cette thèse concerne essentiellement l’agencement statique, et les méthodes proposées ne prennent pas en compte les chemins géométriques permettant d’aboutir aux solutions proposées. L’utilisation d’heuristiques de placement comme l’heuristique *Bottom-Left* qui assurent un placement séquentiel des composants, permet de répondre à la question pour les problèmes de découpe et de conditionnement. Concernant l’agencement, un développement spécifique doit être mené pour apporter des solutions à ce problème ;
- l’interaction des processus d’optimisation avec l’utilisateur. Les interactions entre le concepteur et les algorithmes d’optimisation peuvent être de deux types. Le premier type concerne l’évaluation humaine d’objectifs qu’on ne saurait exprimer mathématiquement. Le second type concerne la modification de solutions par le concepteur. Avec notre méthode, les variables d’optimisation étant directement les variables de positionnement des composants, le concepteur peut facilement interagir avec les solutions générées et proposer des solutions qui lui semblent prometteuses. Pour cela, l’intégration d’un algorithme d’optimisation interactif [Tak00] est nécessaire. Une thèse est en cours à l’IRCCyN pour prendre en compte ces aspects ;
- l’intégration de l’optimisation d’agencement dans les processus de conception. Actuellement, les problèmes de placement sont traités comme des problèmes à part entière. Toutefois, ces problèmes font partie intégrante des problèmes de conception, plus généraux. La résolution simultanée de ces deux problèmes est un problème multidisciplinaire, pour lequel les techniques d’optimisation sont en plein essor. Les premiers travaux dans ce sens concernent l’agencement de composants à l’intérieur d’un châssis de géométrie déformable [DFB06].

Annexe A

Abréviations

Cette annexe présente les différentes abréviations liées aux problèmes de placement et à l’optimisation. Les références bibliographiques permettent d’aller chercher davantage d’informations dans les articles ou ouvrages de référence.

Abréviations	Correspondance	Traduction	Réf.
<i>Problèmes de découpe et de conditionnement</i>			
<i>C&PP</i>	Cutting and Packing problem	Problème de découpe et de conditionnement	
<i>OR</i>	Operational Research	Recherche Opérationnelle	
<i>2BPP</i>	2D Bin Packing Problem	Problème bidimensionnel de conditionnement de rectangles	
<i>CSP</i>	Cutting Stock Problem	Problème de découpe	
<i>MLCSP</i>	Multiple Length Cutting Stock Problem	Problème de découpe unidimensionnelle avec différents stocks	[AdC08]
<i>2OPP</i>	2D Orthogonal Packing Problem	Problème de décision bidimensionnel de conditionnement de rectangles	[CJCM08]
<i>3KCLP</i>	3D Knapsack Container Loading Problem	Problème de chargement de containers	[EP03]
<i>ODP</i>	Open Dimension Problem		[WHS07]
<i>Problèmes d’agencement</i>			
<i>LP</i>	Layout problem	Problème d’agencement	
<i>FLP</i>	Facility Layout Problem	Problème d’agencement de locaux	
<i>VLSI</i>	Very Large Scale Integration	Problème d’intégration à grande échelle	
<i>CDP</i>	Configuration Design Problem	Problème d’agencement en ingénierie	

Table A.1 – Abréviations des problèmes de placement

Abréviations	Correspondance	Traduction	Réf.
<i>Méthodologie développée pour la résolution des problèmes d’agencement</i>			
<i>A-TEAM</i>	Asynchronous-Teams		[SPGT98]
<i>CDOM</i>	Configuration Design Optimization Method		[Gri98]
<i>EPS</i>	Extended Pattern Search		[YC00]
<i>OPS</i>	Objective function effect based Pattern Search		[ACS07b, ACS07a]
<i>MOPS</i>	Metric Objective function effect based Pattern Search		[ACS07a]
<i>HAKD</i>	Human-Algorithm-Knowledge-based layout Design method		[LT08]
<i>Méthode de résolution des problèmes de conditionnement irrégulier</i>			
<i>2D&3D Nest</i>	2D & 3D Nest		[ENO07, ENB09]
<i>BSN</i>	Beam Search for Nesting problems		[BS07]
<i>GLS</i>	Guided Local Search		[FPZ03]
<i>SAHA</i>	Simulated Annealing Hybrid Algorithm		[GO06]
<i>GLSHA</i>	Greedy Local Search Hybrid Algorithm		[GO06]
<i>ILSQN</i>	Iterated Local Search Quasi-Newton algorithm		[IYN09]

Table A.2 – Abréviations des méthodes de résolution des problèmes de placement.

Abréviations	Correspondance	Traduction	Réf.
<i>Heuristiques de placement</i>			
<i>BL</i>	Bottom-Left		[Jak96]
<i>IBL</i>	Improved Bottom-Left		[LT99]
<i>BLF</i>	Bottom-Left Fill		[HT00]
<i>BLBF</i>	Bottom-Left-Back Fill		[TFF08]
<i>MERA</i>	Minimization of Enclosing Rectangle Area		[ABHI05]
<i>MERAG</i>	Minimization of Enclosure under Gravitational Attraction		[ABHI05]
<i>MERAM</i>	Minimization of Enclosure under Magnetic Attraction		[ABHI05]
<i>Schéma d'encodage</i>			
<i>B*-Tree</i>	Binary Tree		[CCWW00]
<i>BSG</i>	Bounded Sliceline Grid		[NFMK96]
<i>CBL</i>	Corner Block List		[HHC+00]
<i>Fast-SP</i>	Fast-Sequence Pair		[TTW00]
<i>IPN</i>	Inverse Polish Notation		[OI98]
<i>O-Tree</i>	Ordered Tree		[GCY99]
<i>SP</i>	Sequence Pair		[MFNK96, Pis07]
<i>TCG</i>	Transitive Closure Graph		[LC01]

Table A.3 – Abréviations des heuristiques de placement et des schémas d'encodage.

Abréviations	Correspondance	Traduction	Réf.
<i>Traitement géométrique</i>			
<i>NFP</i>	No-Fit Polygon	Polygone de non-recouvrement	[BDD01]
<i>IFP</i>	Inner-Fit Polygon	Polygone d'appartenance	[BDD01]
<i>BB</i>	Bounding Box	Boîte englobante	
<i>AABB</i>	Axis Aligned Bounding Box	Boîte englobante alignée sur les axes principaux	
<i>OBB</i>	Oriented Bounding Box	Boîte englobante orientée	
<i>OBB-tree</i>	Oriented Bounding Box Tree	Arbre de boîtes englobantes orientées	[GLM96]
<i>BSP</i>	Binary Space Partitioning	Partitionnement binaire de l'espace	[TN87]
<i>Sphere-tree</i>	Sphere tree	Arbre de sphères	[PG95, Hub96]
<i>Octree</i>	Octant tree	Arbre d'octants	
<i>Voxel</i>	Volumetric pixel	Volume élémentaire	
<i>k-DOP</i>	Discrete Orientation Polytope	Polytope à orientations discrètes	[KHM+98]

Table A.4 – Abréviations des outils et représentations géométriques.

Abréviations	Correspondance	Traduction	Réf.
<i>Les différents types de problèmes</i>			
<i>IP</i>	Integer Program	Programmation Linéaire en Nombres Entiers (PLNE)	
<i>MIP</i>	Mixed Integer Program	Programmation Linéaire Mixte (PLM)	
<i>NLP</i>	NonLinear Program	Programmation Non Linéaire (PNL)	
<i>MNLP</i>	Mixed NonLinear Program	Programmation Non Linéaire Mixte (PNLM)	
<i>MOOP</i>	Multi-Objective Optimization Problem	Problème d'optimisation multi-objectif	

Table A.5 – Abréviations des différents types de problèmes d'optimisation rencontrés.

Abréviations	Correspondance	Traduction	Réf.
<i>Heuristiques</i>			
<i>ACO</i>	Ant Colony Optimisation	Optimisation par colonie de fourmis	
<i>SA</i>	Simulated Annealing	Recuit simulé	[KGV83]
<i>MOSA</i>	Multi-Objective Simulated Annealing	Recuit simulé multi-objectif	
<i>PSO</i>	Particle Swarm Optimization	Optimisation par essaim de particules	
<i>MOPSO</i>	Multi-Objective Particle Swarm Optimization	Optimisation par essaim de particules multi-objectif	
<i>Méthodes de recherche directe</i>			
<i>DS</i>	Direct Search	Méthode de recherche directe	
<i>GPS</i>	Generalized Pattern Search	Méthode de recherche généralisée par motifs	[TT97]

Table A.6 – Abréviations des différentes heuristiques et métaheuristiques.

Abréviations	Correspondance	Traduction	Réf.
<i>Algorithmes évolutionnaires</i>			
EA	Evolutionary Algorithm	Algorithme évolutionnaire	
DE	Differential Evolution	Évolution différentielle	
EP	Evolutionary Programming	Programmation évolutionniste	[Fog62, Fog88]
ES	Evolution Strategy	Stratégie d'évolution	
GP	Genetic Programming	Programmation génétique	
HEA	Hybrid Evolutionary Algorithm	Algorithme évolutionnaire hybride	
GA	Genetic Algorithm	Algorithme génétique	[Gol89]
GGA	Generational Genetic Algorithm	Algorithme génétique générationnel	
SSGA	Steady-State Genetic Algorithm	Algorithme génétique à génération continue	
MOEA	MultiObjective Evolutionary Algorithm	Algorithme évolutionnaire multi-objectifs	
<i>Algorithmes évolutionnaires multi-objectifs (Algorithmes génétiques et stratégies évolutionnaires)</i>			
AMGA	Archive-based Micro Genetic Algorithm		[TFKD08, TFKD09]
AMGA2	Archive-based Micro Genetic Algorithm 2		[TFD09]
DMOEA	Dynamic Multi-Objective Evolutionary Algorithm		[YL03]
C-NSGA-II	Clustered Non-dominated Sorting Genetic Algorithm		[DMM03]
CMA-ES	Covariance Matrix Adaptation Evolution Strategies		[HMK03, HK04]
FastPGA	Fast Pareto Genetic Algorithm		[KL05]
GDE3	Generalized Differential Evolution		[EGL07]
hBOA	hierarchical Bayesian Optimization Algorithm		[PSG05]
HypE	Hypervolume Estimation algorithm for MO		[BZ08]
IBEA	Indicator Based Evolutionary Algorithm		[ZK04]
IMOEa	Intelligent Multi-Objective Evolutionary Algorithm		[HSC04]
mBOA	multi-objective Bayesian Optimization Algorithm		[KGP02]
ε -MOEA	ε -domination Multi-Objective Evolutionary Algorithm		[DMM05]
MOGA	Multi-Objective Genetic Algorithm		[FF93]
MOGA-II	Multi-Objective Genetic Algorithm - II		[MP93, PP98]
Omni-Optimizer	Omni-Optimizer		[DT08]
NCGA	Neighborhood Cultivation Genetic Algorithm		[WHM02]
NPGA	Niched Pareto Genetic Algorithm		[HNG94]
NSGA	Non-dominated Sorting Genetic Algorithm		[SD94]
NSGA-II	Elitist Non-dominated Sorting Genetic Algorithm		[DAPM00b]
PAES	Pareto-Archived Evolution Strategy		[KC00]
PESA	Pareto-Envelope based Selection Algorithm		[CKO00]
PESA-II	Pareto-Envelope based Selection Algorithm 2		[CJKO01]
SPEA	Strength Pareto Evolutionary Algorithm		[ZT98]
SPEA2	Strength Pareto Evolutionary Algorithm 2		[ZLT01]
SPEA2+	Improved Strength Pareto Evolutionary Algorithm 2		[KHMW04]
RWGA	Random-Weight based Genetic Algorithm		[MI95]
VEGA	Vector Evaluated Genetic Algorithm		[Sch84]
VOES	Vector-Optimized Evolution Strategy		[Kur90]

Table A.7 – Abréviations de différents algorithmes évolutionnaires.

Annexe B

Outils pour la résolution des problèmes de placement

Ce chapitre présente les différents outils utilisés pour résoudre les problèmes de placement.

Sommaire

B.1	Introduction	146
B.2	Optimisation continue sans contrainte	146
B.3	Optimisation multi-objectif	148
B.3.1	Procédures de l'algorithme <i>Omni-Optimizer</i>	148
B.3.2	Détails de certains opérateurs continus	153
B.4	Polygones de non-recouvrement et polygones d'appartenance	155
B.4.1	Polygones de non-recouvrement	156
B.4.2	Polygones d'appartenance	159
B.4.3	Remarques générales	159
B.5	Angle d'Euler	159

B.1 Introduction

Les différents outils algorithmiques et géométriques utilisés pour la résolution des problèmes de placement sont présentés dans ce chapitre. Les deux premières sections concernent l'optimisation. La section B.2 présente l'algorithme d'optimisation continue *BFGS* utilisé pour effectuer les séparations de composants. La section B.3 contient les routines de l'algorithme d'optimisation globale *Omni-Optimizer* avec des détails sur les opérateurs génétiques utilisés. La section B.4 explique le fonctionnement des polygones de non-recouvrement et d'appartenance, utilisé pour détecter les collisions entre polygones. Les techniques de calculs de ces polygones sont aussi présentées ainsi que leurs différentes utilisations. Enfin, les angles d'Euler utilisés pour représenter l'orientation d'un composant en 3D sont présentés section B.5.

B.2 Optimisation continue sans contrainte

On cherche à minimiser une fonction continue $f : \mathbb{R}^n \rightarrow \mathbb{R}$

$$\mathcal{P} : \min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \quad (\text{B.1})$$

Une condition nécessaire pour que le vecteur \mathbf{x}^* soit solution du problème d'optimisation $\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$ avec $f : \mathbb{R}^n \rightarrow \mathbb{R}$ est qu'il annule le gradient de f :

$$\nabla f(\mathbf{x}^*) = \mathbf{0} \quad (\text{B.2})$$

Pour résoudre cette équation bien souvent non linéaire, la méthode de Newton-Raphson peut être utilisée. Cette méthode itérative permet de trouver une solution au problème $f(\mathbf{x}) = 0$ à partir d'un point initial $\mathbf{x}_0 \in \mathbb{R}^n$. Dans le cadre de la méthode de placement proposée, ce point initial est fourni par l'optimiseur global, à savoir l'algorithme génétique. Si la fonction f possède plusieurs racines, on peut envisager de tester plusieurs solutions initiales pour trouver l'ensemble des racines de la fonction f . La méthode de Newton est basée sur le développement limité de la fonction autour d'un point \mathbf{x}_k .

On considère le cas d'une fonction mono-variable. Le développement limité de cette fonction s'écrit :

$$f(x_k + \Delta x) = f(x_k) + f'(x_k)\Delta x + \frac{1}{2}f''(x_k)\Delta x^2 + O(x)^2 \quad (\text{B.3})$$

En ne considérant uniquement le développement à l'ordre premier, on peut approximer la fonction f par sa tangente. Avec une solution initiale $x_0 \in \mathbb{R}$, on peut construire itérativement les x_k successifs pour résoudre l'équation $f(x) = 0$:

$$x_{k+1} = x_k - f'(x_k)^{-1}f(x_k) \quad (\text{B.4})$$

La figure B.1 illustre le processus de convergence de cette méthode. En appliquant le principe de résolution de la méthode de Newton-Raphson à l'équation $f'(x) = 0$, on peut identifier le minimum local de f le plus proche d'un point initial x_0 . On obtient alors :

$$f'(x) + f''(x)\Delta x = 0 \quad (\text{B.5})$$

Le processus itératif de résolution s'écrit alors :

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}, \quad k \geq 0 \quad (\text{B.6})$$

Ce schéma peut être généralisé à plusieurs dimensions en remplaçant la dérivée f' par le gradient $\nabla f(\mathbf{x})$, et la dérivée seconde f'' par la matrice hessienne $\mathbf{H}(\mathbf{x})$. Le schéma itératif de résolution devient

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{H}_f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k), \quad k \geq 0 \quad (\text{B.7})$$

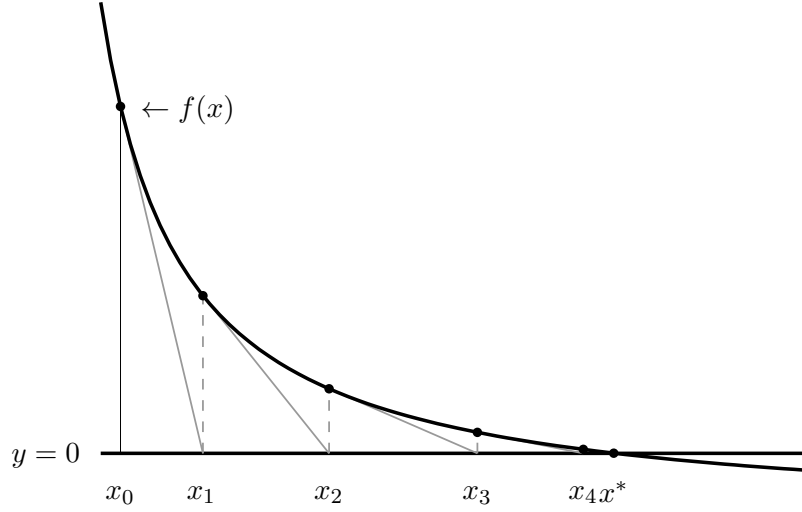


Figure B.1 – Illustration de la convergence de la méthode de Newton-Raphson pour la fonction $f(x) = 1/x - 1/3$ avec $x_0 = 0.5$. La solution x^* est bien évidemment obtenue pour $x = 3$.

Un coefficient $\gamma > 0$ peut être utilisé pour ajuster la correction effectuée.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma [\mathbf{H}_f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k), \quad k \geq 0 \quad (\text{B.8})$$

Dans le cas où l'expression analytique de la matrice hessienne $\mathbf{H}(\mathbf{x})$ n'est pas disponible, ou si l'inversion de la matrice hessienne est trop coûteuse ($\mathcal{O}(n^3)$), une approximation de l'inverse de la matrice hessienne peut être calculée par les méthodes dites *quasi-Newton*.

Parmi les méthodes *quasi-Newton*, on peut citer :

- la méthode SR1 (**S**ymmetric **R**ank **1**) qui effectue une mise à jour de rang 1 sur la matrice hessienne (La correction appliquée sur la matrice hessienne est une matrice de rang 1) [BKS96] ;
- la formule de *Davidon, Fletcher, Powell* (DFP), qui utilise une correction de rang 2 pour la mise à jour de la hessienne [Dav59, FP63] ;
- la méthode de *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) [Bro70, Fle70, Gol70, Sha70], qui est aussi une méthode de rang 2 et qui est accessoirement la plus populaire.

Présentée dans l'algorithme B.1, la méthode *BFGS* réactualise la matrice hessienne entre deux itérations de la manière suivante :

$$\mathbf{H}_k = \mathbf{H}_{k-1} + \frac{\mathbf{q}_k \mathbf{q}_k^t}{\mathbf{q}_k^t \mathbf{s}_k} - \frac{\mathbf{H}_{k-1}^t \mathbf{s}_k^t \mathbf{s}_k \mathbf{H}_{k-1}}{\mathbf{s}_k^t \mathbf{H}_{k-1} \mathbf{s}_k} \quad (\text{B.9})$$

avec $\mathbf{s}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$ et $\mathbf{q}_k = \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1})$. Le calcul de l'inverse de la matrice hessienne se fait à l'aide de la formule de *Sherman-Morrison-Woodbury* :

$$\mathbf{H}_k^{-1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{q}_k^t}{\mathbf{s}_k^t \mathbf{q}_k} \right) \mathbf{H}_{k-1}^{-1} \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{q}_k^t}{\mathbf{s}_k^t \mathbf{q}_k} \right) + \frac{\mathbf{s}_k \mathbf{q}_k^t}{\mathbf{s}_k^t \mathbf{q}_k} \quad (\text{B.10})$$

où \mathbf{I} désigne la matrice identité. L'algorithme *L-BFGS* [LN89] correspond à la version *BFGS* pour les problèmes de grande taille. Cet algorithme a par la suite été amélioré pour prendre en compte d'éventuelles bornes sur les variables [BLNZ95, BNZ97].

Algorithme B.1: Algorithme quasi-Newton *BFGS*

Données : Fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}$ continument différentiable, Gradient de la fonction $\nabla f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, Solution initiale $\mathbf{x}_0 \in \mathbb{R}^n$, Tolérance ε , Nombre maxi itérations k_{\max} **Résultat :** Vecteur solution $\mathbf{x}^* \in \mathbb{R}^n$

```
1 Initialisation
2  $k \leftarrow 0$  // Compteur d'itérations ;
3  $\alpha_0 \leftarrow 1$  // Initialisation du coefficient  $\alpha$ 
4  $\mathbf{H}_0^{-1} \leftarrow \mathbf{I}$  // Initialisation de l'inverse de la hessienne
5 Itérations
6 tant que ( $k < k_{\max}$   $\mathcal{E} \|\nabla f(\mathbf{x}_k)\| \geq \varepsilon$ ) faire
    1.  $\mathbf{d}_k \leftarrow -\mathbf{H}_k^{-1} \nabla f(\mathbf{x}_k)$ 
    2. Déterminer  $\alpha_k$  tel que  $f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)$  soit minimum (Algorithme de recherche linéaire) ;
    3.  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$ 
    4.  $k \leftarrow k + 1$ 
    5.  $\mathbf{s}_k \leftarrow \mathbf{x}_k - \mathbf{x}_{k-1}$  ;  $\bar{\mathbf{s}}_k \leftarrow \alpha_k \mathbf{s}_k$ 
    6.  $\mathbf{q}_k \leftarrow \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1})$ 
    7. Mettre à jour  $\mathbf{H}_k^{-1}$ 
        
$$\mathbf{H}_k^{-1} \leftarrow \left( \mathbf{I} - \frac{\bar{\mathbf{s}}_k \mathbf{q}_k^t}{\bar{\mathbf{s}}_k^t \mathbf{q}_k} \right) \mathbf{H}_{k-1}^{-1} \left( \mathbf{I} - \frac{\bar{\mathbf{s}}_k \mathbf{q}_k^t}{\bar{\mathbf{s}}_k^t \mathbf{q}_k} \right) + \frac{\bar{\mathbf{s}}_k \mathbf{q}_k^t}{\bar{\mathbf{s}}_k^t \mathbf{q}_k}$$

7 fin
```

B.3 Optimisation multi-objectif

B.3.1 Procédures de l'algorithme *Omni-Optimizer*

Les algorithmes B.2, B.3, B.4, B.5, B.6, B.7, présentent les différentes étapes de l'algorithme *Omni-Optimizer*. L'algorithme B.2 présente le déroulement global de *Omni-Optimizer*. Les pseudo-codes des opérations de sélection par tournoi mono- et multi-objectif sont présentés dans les algorithmes B.3, B.4. La procédure de sélection des deux individus proches dans l'espace des variables est présentée dans l'algorithme B.5. Le tri multi-objectif des solutions ainsi que le calcul des distances entre solutions appartenant au même front sont contenus dans les algorithmes B.6, B.7.

Algorithme B.2: Pseudo-code de l'algorithme *Omni-Optimizer***Omni-Optimizer**

```

1   $t = 0$       (Compteur de générations)
2  Générer la population initiale  $P_0$  comportant  $N$  individus
3  Évaluer  $P_0$ 
4  répéter
5      Créer une population intermédiaire non triée  $R_t$  à l'aide de la population parent  $P_t$ 
6       $R_t \leftarrow \text{Shuffle}(P_t) \cup \text{Shuffle}(P_t)$ 
7      Générer la population enfant  $Q_t$  à partir de la population intermédiaire  $R_t$ 
8      pour  $i = 1 : 2 : N - 1$  faire
9          Effectuer les opérations de sélection
10         (player1,player2)  $\leftarrow$  choose_nearest( $R_t$ )
11         parent1  $\leftarrow$  tournament(player1,player2)
12         (player1,player2)  $\leftarrow$  choose_nearest( $R_t$ )
13         parent2  $\leftarrow$  tournament(player1,player2)
14         Effectuer les opérations de croisement et de mutation
15         (offspring1,offspring2)  $\leftarrow$  variation(parent1,parent2)
16          $Q_t(i) \leftarrow$  offspring1
17          $Q_t(i+1) \leftarrow$  offspring2
18          $i \leftarrow i + 2$ 
19      fin
20      Évaluer la population enfant  $Q_t$ 
21      Fusionner les populations parent et enfant
22       $R_t \leftarrow P_t \cup Q_t$ 
23      Calculer le rang de chaque solution de  $R_t$ 
24       $(F_1, F_2, \dots) \leftarrow \text{ranking}(R_t)$ 
25      Initialiser la prochaine population
26       $P_{t+1} \leftarrow \emptyset$ 
27      Copier les solutions des meilleurs fronts  $F_j$  dans la population  $P_{t+1}$ 
28       $j \leftarrow 1$ 
29      tant que  $|P_{t+1} \cup F_j| \leq N$  faire
30           $P_{t+1} \leftarrow P_{t+1} \cup F_j$ 
31          Effectuer les calculs de distances génotypiques et phénotypiques pour
32          les solutions du front  $F_j$ 
33          crowd_dist( $F_j$ )
34           $j \leftarrow j + 1$ 
35      fin
36       $L \leftarrow j$ 
37      Calculer le nombre de solutions du front  $F_L$  à ajouter à  $P_{t+1}$ 
38       $rem \leftarrow N - |P_{t+1}|$ 
39      Trier le front  $F_L$  dans l'ordre décroissant des distances des solutions
40      sorting(crowd_dist( $F_L$ ))
41      Compléter la prochaine population  $P_{t+1}$  avec les  $rem$  premiers individus du front  $F_L$ 
42       $P_{t+1} \leftarrow P_{t+1} \cup F_L(1 : rem)$ 
43       $t \leftarrow t + 1$ 
44  jusqu'à Validation d'un critère d'arrêt

```

Algorithme B.3: Méthode de sélection par tournoi pour un problème d'optimisation mono-objectif sous contraintes. $CV(a)$ et $CV(b)$ sont les indices de violation de contraintes des solutions a et b . Les distances génotypique et/ou phénotypique sont regroupées dans l'expression $crowd_dist$.

```
c  $\leftarrow$  Binary_tournament_selection_SO(a,b)
Données : Individus a et b
Résultat : Individu sélectionné c
1 si a vérifie les contraintes mais pas b, alors c  $\leftarrow$  a
2 sinon si b vérifie les contraintes mais pas a, alors c  $\leftarrow$  b
3 sinon si a et b ne vérifient pas les contraintes, alors
4   | si  $CV(a) < CV(b)$  alors c  $\leftarrow$  a
5   | sinon c  $\leftarrow$  b
6 fin
7 sinon si a et b vérifient les contraintes, alors
8   | si  $f(a) < f(b)$ , alors c  $\leftarrow$  a
9   | sinon si  $f(b) < f(a)$ , alors c  $\leftarrow$  b
10  | sinon si  $crowd\_dist(a) > crowd\_dist(b)$ , alors c  $\leftarrow$  a
11  | sinon si  $crowd\_dist(a) < crowd\_dist(b)$ , alors c  $\leftarrow$  b
12  | sinon c  $\leftarrow$  random_choose(a, b)
13 fin
```

Algorithme B.4: Méthode de sélection par tournoi pour un problème d'optimisation multi-objectif sous contraintes. $CV(a)$ et $CV(b)$ sont les indices de violation de contraintes des solutions a et b . Les distances génotypique et/ou phénotypique sont regroupées dans l'expression $crowd_dist$.

```
c  $\leftarrow$  Binary_tournament_selection_MO(a,b)
Données : Individus a et b
Résultat : Individu sélectionné c
1 si a vérifie les contraintes mais pas b, alors c  $\leftarrow$  a
2 sinon si a et b ne vérifient pas les contraintes, alors
3   | si  $CV(a) < CV(b)$  alors c  $\leftarrow$  a
4   | sinon c  $\leftarrow$  b
5 fin
6 sinon si a et b vérifient les contraintes, alors
7   | si a domine b, alors c  $\leftarrow$  a
8   | sinon si b domine a, alors c  $\leftarrow$  b
9   | sinon si  $crowd\_dist(a) > crowd\_dist(b)$ , alors c  $\leftarrow$  a
10  | sinon si  $crowd\_dist(a) < crowd\_dist(b)$ , alors c  $\leftarrow$  b
11  | sinon c  $\leftarrow$  random_choose(a, b)
12 fin
```

Algorithme B.5: Procédure de sélection de deux individus proches dans l'espace des variables. Le calcul de la distance génotypique entre deux solutions dépend des variables de chaque problème.

```

  ( $\mathbf{a}, \mathbf{b}$ )  $\leftarrow$  choose_nearest( $R_t$ )
  Données : Population  $R_t$ 
  Résultat : Individus sélectionnés  $\mathbf{a}$  &  $\mathbf{b}$ 
1  Sélectionner le premier individu de  $R_t$ , et le supprimer de la population  $R_t$ 
2   $\mathbf{a} \leftarrow \text{pop}(R_t(1))$ 
3   $\text{dist} \leftarrow \infty$ 
4  pour  $i \leftarrow 2$  à  $|R_t|$  faire
5    Calculer la distance génotypique entre l'individu  $\mathbf{a}$  et  $R_t(i)$ 
6     $\text{temp} \leftarrow \text{compute\_normalized\_genotypic\_distance}(\mathbf{a}, R_t(i))$ 
7    si  $\text{temp} < \text{dist}$  alors
8       $\text{index} \leftarrow i$ 
9       $\text{dist} \leftarrow \text{temp}$ 
10   fin
11 fin
12 Sélectionner l'individu  $\mathbf{b}$  le plus proche de  $\mathbf{a}$  et le supprimer de la population  $R_t$ 
13  $\mathbf{b} \leftarrow \text{pop}(R_t(\text{index}))$ 

```

Algorithme B.6: Procédure de classement des solutions *ranking*.

```

  ( $F_1, F_2, \dots$ )  $\leftarrow$  ranking( $R_t$ )
  Données : Population  $R_t$  contenant  $2N$  solutions
  Résultat : Solutions triées selon les fronts ( $F_1, F_2, \dots$ )
1   $k \leftarrow 1$ 
2  répéter
3     $F_k \leftarrow \emptyset$ 
4    pour  $i \leftarrow 1$  à  $|R_t|$  faire
5      pour  $j \in \{1, \dots, |R_t|\} \setminus i$  faire
6        si  $R_t(j) \varepsilon\text{-domine } R_t(i)$  alors arrêt
7      fin
8      si  $j = |R_t|$  alors  $F_k \leftarrow F_k \cup R_t(i)$ 
9    fin
10    $R_t \leftarrow R_t \setminus F_k$ 
11    $k \leftarrow k + 1$ 
12 jusqu'à ce que les  $2N$  solutions soient classées

```

Algorithme B.7: Procédure de calcul des distances entre solutions d'un même front.

```
crow_dist  $\leftarrow$  Compute_crowding_distance( $F_j$ )
Données : Solutions du front  $F_j$ 
Résultat : crow_dist : Distance entre les solutions du front  $F_j$ 
1 Initialisation des distances phénotypique et génotypique pour les  $|F_j|$  solutions du front  $F_j$ 
2 pour  $i \leftarrow 1$  à  $|F_j|$  faire
3   | crow_dist_obj( $i$ )  $\leftarrow 0$ 
4   | crow_dist_var( $i$ )  $\leftarrow 0$ 
5 fin
6 Calcul des distances phénotypiques
7 pour  $m \leftarrow 1$  à  $p$  faire
8   | pour  $i \leftarrow 1$  à  $|F_j|$  faire
9     | si  $i$  est une solution extrême pour le  $m^{i\text{ème}}$  objectif alors
10    | | crow_dist_obj( $i$ )  $\leftarrow \infty$ 
11    | sinon
12    | | crow_dist_obj( $i$ )  $\leftarrow$  crow_dist_obj( $i$ ) + normalized_obj( $i$ )
13    | fin
14  | fin
15 fin
16 Calcul des distances génotypiques
17 pour  $j \leftarrow 1$  à  $n$  faire
18   | pour  $i \leftarrow 1$  à  $|F_j|$  faire
19     | si  $i$  est une solution extrême pour la  $j^{i\text{ème}}$  variable alors
20     | | crow_dist_var( $i$ )  $\leftarrow$  crow_dist_var( $i$ ) +  $2 \times$  normalized_var( $i$ )
21     | sinon
22     | | crow_dist_var( $i$ )  $\leftarrow$  crow_dist_var( $i$ ) + normalized_var( $i$ )
23     | fin
24   | fin
25 fin
26 Normalisation des distances
27 pour  $i \leftarrow 1$  à  $|F_j|$  faire
28   | crow_dist_obj( $i$ )  $\leftarrow$  crow_dist_obj( $i$ ) /  $p$ 
29   | crow_dist_var( $i$ )  $\leftarrow$  crow_dist_var( $i$ ) /  $n$ 
30 fin
31 Calcul des distances moyennes
32 avg_crow_dist_obj  $\leftarrow \sum_{i=1:|F_j|} \text{crow\_dist\_obj}(i) / |F_j|$ 
33 avg_crow_dist_var  $\leftarrow \sum_{i=1:|F_j|} \text{crow\_dist\_var}(i) / |F_j|$ 
34 pour  $i \leftarrow 1$  à  $|F_j|$  faire
35   | si ( $\text{crow\_dist\_obj}(i) > \text{avg\_crow\_dist\_obj}$ )  $\vee$  ( $\text{crow\_dist\_var}(i) > \text{avg\_crow\_dist\_var}$ )
36   | | alors
37   | | | crow_dist( $i$ )  $\leftarrow \max \{ \text{crow\_dist\_obj}(i), \text{crow\_dist\_var}(i) \}$ 
38   | | sinon
39   | | | crow_dist( $i$ )  $\leftarrow \min \{ \text{crow\_dist\_obj}(i), \text{crow\_dist\_var}(i) \}$ 
40   | fin
41 fin
```

B.3.2 Détails de certains opérateurs continus

Lors de la résolution de problèmes avec des variables continues, il n'est pas nécessaire de discrétiser les domaines de variation de ces variables. Il est possible de les traiter entièrement de manière continue. Le génotype fournit directement les valeurs des différentes variables, les opérations d'encodage / décodage sont inexistantes. Toutefois, cela nécessite des opérateurs génétiques adaptés aux variables continues. Plusieurs opérateurs ont été proposés, les plus connus / employés sont ici présentés, à savoir l'opérateur de croisement *SBX* et l'opérateur de mutation polynomiale.

B.3.2.1 Croisement continu *SBX*

Introduit par Deb [DA95, DB99], l'opérateur de croisement continu *Simulated Binary Crossover* (*SBX*) permet de générer des enfants proches de l'un des parents. Soient x_{p_1} et x_{p_2} les valeurs réelles des individus parents. L'objectif du croisement est de générer des individus enfant x_{c_1} et x_{c_2} à partir de deux parents x_{p_1} et x_{p_2} .

$$\begin{aligned} x_{c_1} &= \frac{1}{2} [(x_{p_1} + x_{p_2}) - \beta_q |x_{p_1} - x_{p_2}|] \\ x_{c_2} &= \frac{1}{2} [(x_{p_1} + x_{p_2}) + \beta_q |x_{p_1} - x_{p_2}|] \end{aligned} \quad (\text{B.11})$$

avec β_q un nombre issu de la distribution polynomiale suivante :

$$\beta_q = \begin{cases} (2r)^{\frac{1}{\eta_c+1}} & \text{si } r \leq 0.5 \\ \left(\frac{1}{2(1-r)}\right)^{\frac{1}{\eta_c+1}} & \text{si } r > 0.5 \end{cases} \quad (\text{B.12})$$

Le paramètre r est un nombre aléatoire distribué de manière uniforme entre 0 et 1. Le paramètre η_c permet de contrôler la distribution des solutions générées. Une faible valeur de η_c permet de générer des individus enfants éloignés des solutions parents, et *vice versa*. La figure B.2 illustre cette influence du paramètre η_c sur la génération des individus enfants. Les individus enfants sont symétriques par rapport à leurs parents de par la structure de la formule B.11. Dans le cas où les variables sont bornées entre x^l et x^u , l'expression β_q devient [DA99] :

$$\beta_q = \begin{cases} (r\alpha)^{\frac{1}{\eta_c+1}} & \text{si } r \leq \frac{1}{\alpha} \\ \left(\frac{1}{2-r\alpha}\right)^{\frac{1}{\eta_c+1}} & \text{si } r > \frac{1}{\alpha} \end{cases} \quad (\text{B.13})$$

où $\alpha = 2 - \beta^{-(\eta_c+1)}$, $\beta = 1 + \frac{2}{|x_{p_2} - x_{p_1}|} \min \left\{ \left(\min \{x_{p_1}, x_{p_2}\} - x^l \right), (x^u - \max \{x_{p_1}, x_{p_2}\}) \right\}$, et r est un nombre aléatoire uniformément distribué entre 0 et 1.

B.3.2.2 Mutation polynomiale

La distribution de probabilité est polynomiale [DA95]. Un enfant x_c est généré à partir d'un parent x_p à l'aide de la relation suivante

$$x_c = x_p + \Delta_{\max} \delta_q \quad (\text{B.14})$$

où Δ_{\max} correspond à la plage de variation autorisée et δ_q est le coefficient de variation calculé à partir de la distribution polynomiale : $P(\delta) = 0.5(\eta_m + 1)(1 - |\delta|)^{\eta_m}$. La figure B.3 illustre cette répartition de probabilité pour différentes valeurs du paramètre η_m . Dans le cas où les variables ne sont pas bornées, δ_q est évalué de la manière suivante :

$$\delta_q = \begin{cases} (2r)^{\frac{1}{\eta_m+1}} - 1 & \text{si } r \leq 0.5 \\ 1 - [2(1-r)]^{\frac{1}{\eta_m+1}} & \text{si } r > 0.5 \end{cases} \quad (\text{B.15})$$

où r est un nombre aléatoire uniformément distribué dans l'intervalle $[0; 1]$, η_m est l'indice de distribution de la mutation. Si les variables sont bornées, $\Delta_{\max} = (x^u - x^l)$, où x^l et x^u sont les bornes inférieures et supérieures du parent à muter. L'opération de mutation devient alors :

$$x_c = x_p + (x^u - x^l) \delta_q \quad (\text{B.16})$$

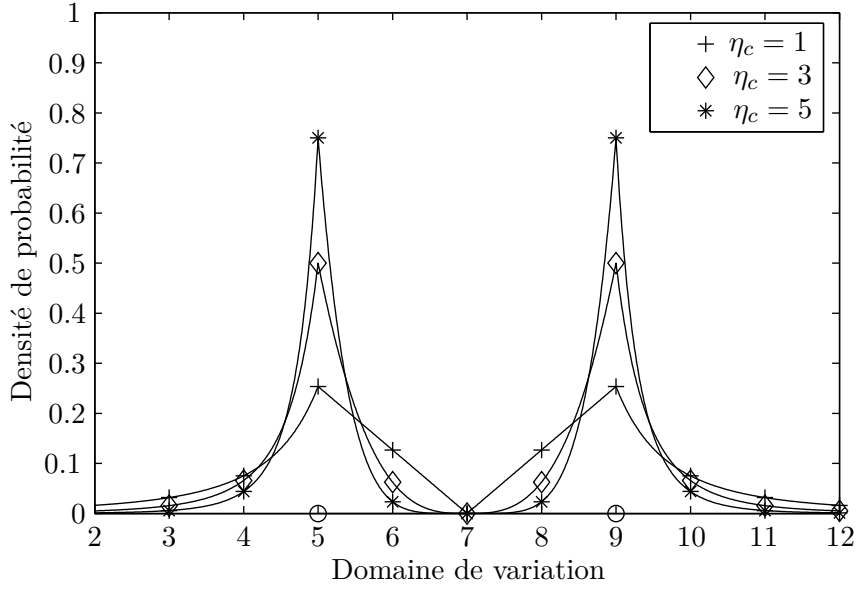


Figure B.2 – Densité de probabilité de création d'enfants avec l'opérateur de croisement SBX- η_c . Les parents sont représentés par 2 cercles sur l'axe des abscisses : $x_{p1} = 5$, $x_{p2} = 9$.

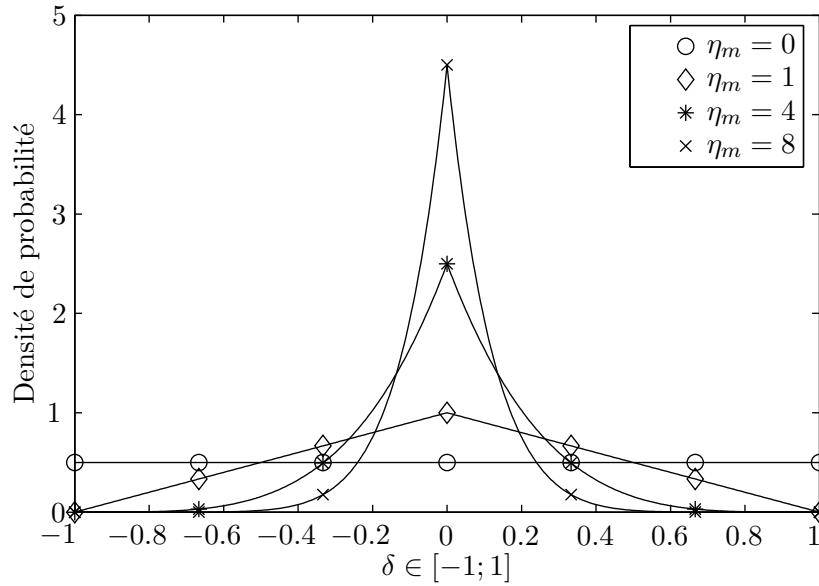


Figure B.3 – Densité de probabilité pour l'opérateur de mutation polynomiale.

La perturbation δ_q est calculée de la manière suivante :

$$\delta_q = \begin{cases} \left[2r + (1 - 2r)(1 - \delta)^{\eta_m + 1} \right]^{\frac{1}{\eta_m + 1}} - 1 & \text{si } r \leq 0.5, \\ 1 - \left[2(1 - r) + 2(r - 0.5)(1 - \delta)^{\eta_m + 1} \right]^{\frac{1}{\eta_m + 1}} & \text{si } r > 0.5 \end{cases} \quad (\text{B.17})$$

Le paramètre δ de l'équation B.17 est calculé de la manière suivante :

$$\delta = \min \left\{ (x_p - x^l), (x^u - x_p) \right\} / (x^u - x^l) \quad (\text{B.18})$$

L'opérateur de mutation polynomiale a par la suite été modifié pour pallier un de ces défauts [DT08] : en effet, lorsque la variable à muter est située sur une de ses bornes, la mutation ne produit aucune

modification. Cela peut conduire à une convergence prématurée. La variation δ_q a donc été corrigée selon

$$\delta_q = \begin{cases} \left[2r + (1 - 2r) \delta_2^{(\eta_m+1)} \right]^{\frac{1}{(\eta_m+1)}} - 1 & \text{si } r \leq 0.5, \\ 1 - \left[2(1 - r) + 2(r - 0.5) \delta_1^{(\eta_m+1)} \right]^{\frac{1}{(\eta_m+1)}} & \text{si } r > 0.5 \end{cases} \quad (\text{B.19})$$

avec

$$\begin{aligned} \delta_1 &= \frac{x_p - x^l}{x^u - x^l} \\ \delta_2 &= \frac{x^u - x_p}{x^u - x^l} = 1 - \delta_1 \end{aligned} \quad (\text{B.20})$$

Pour l'opérateur de mutation polynomiale originale, $\delta = \min\{\delta_1, \delta_2\}$ est utilisé à la place de δ_1 et δ_2 dans l'équation B.19. L'opérateur modifié de mutation polynomiale utilise deux distributions de probabilités différentes dans deux régions différentes de l'espace de recherche, affectant ainsi une probabilité non nulle de générer un individu dans l'espace entier de recherche, et ce même si la variable du parent est située sur une des bornes.

B.4 Polygones de non-recouvrement et polygones d'appartenance

Les polygones de non-recouvrement et d'appartenance (*No-Fit Polygon*, *NFP* et *Inner-Fit Polygon*, *IFP*) sont respectivement utilisés pour détecter le chevauchement entre deux polygones ainsi que l'appartenance d'un polygone à un autre polygone plus grand. Les informations qu'ils fournissent sont également utilisées pour réaliser la séparation en translation de polygones (Voir section 4.3.2).

Basée sur des calculs trigonométriques, la détection classique de chevauchement ou d'appartenance entre deux polygones présente une complexité en $O(mn + m + n)$, où m et n désignent respectivement le nombre de segments des deux polygones [BS08]. L'utilisation des polygones de non-recouvrement et d'appartenance permet de faire tomber cette complexité à $O(k)$, où k représente le nombre de segments du polygone de non-recouvrement ou d'appartenance. De plus, ces polygones présentent l'avantage d'être calculés une seule fois pour un nombre illimité d'utilisations.

La figure B.4 présente un exemple de calcul de ces polygones pour deux polygones P_1 et P_2 . Le polygone P_1 est considéré fixe, le polygone P_2 est considéré glissant. La position de P_2 est repérée à l'aide d'un point de référence, représenté par le cercle noir sur la figure B.4. Les polygones de non-recouvrement et d'appartenance seront définis par rapport à ce point de référence. À gauche, le polygone de non-recouvrement définit l'ensemble des lieux tel que les deux polygones P_1 et P_2 soient en contact avec P_2 à l'extérieur de P_1 . Si le point de référence de P_2 est contenu à l'intérieur de ce polygone, alors il y a collision entre P_1 et P_2 . À droite, le polygone d'appartenance définit l'ensemble des lieux tel que les deux polygones P_1 et P_2 soient en contact avec P_2 entièrement contenu dans P_1 . Si le point de référence de P_2 est situé à l'extérieur de ce polygone, alors P_2 n'est pas intégralement contenu dans P_1 . Dans le cas général, le test d'appartenance d'un point à un polygone consiste à calculer le nombre d'intersections d'une demi-droite infinie partant du point à traiter avec les segments du polygone. Si ce nombre est impair, alors le point est contenu à l'intérieur du polygone. Dans le cas contraire, il est à l'extérieur.

Les termes de polygone de non-recouvrement et polygone d'appartenance représentent un abus de langage : en effet, le résultat de ces calculs géométriques peut produire un ensemble de polygones ainsi qu'un ensemble de segments et de points. Ces deux derniers ensembles représentent des singularités, et demandent une attention particulière pour pouvoir être obtenus [BHKW07]. Pour la simplicité d'écriture, les auteurs s'autorisent cet abus de langage. Enfin, le choix des points de référence des polygones est arbitraire. Une translation du point de référence entraîne une translation identique des polygones de non-recouvrement et d'appartenance.

Les paragraphes suivants présentent les techniques de calcul de ces polygones.

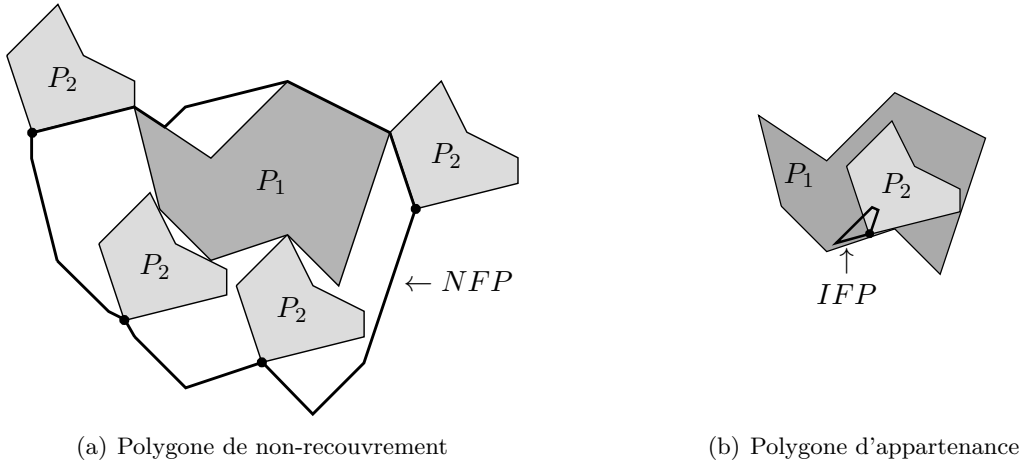


Figure B.4 – Calcul des polygones de non-recouvrement et d'appartenance.

B.4.1 Polygones de non-recouvrement

Les polygones de non-recouvrement sont basés sur l'utilisation de la somme de Minkowski, qui exprime une somme sur deux ensembles. La somme de Minkowski de deux ensembles A et B est définie comme

$$A \oplus B = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in B\} \quad (\text{B.21})$$

Dans la littérature, on peut aussi trouver cette somme sous la forme

$$A \oplus B = \bigcup_{b \in B} A^b \quad (\text{B.22})$$

avec $A^b = \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in A\}$. La figure B.5 montre la somme de Minkowski de l'intérieur de deux polygones.

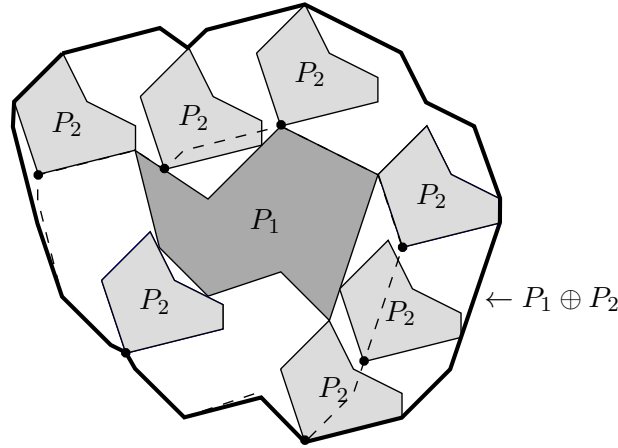


Figure B.5 – Exemple de calcul 2D de somme de Minkowski entre deux polygones P_1 et P_2 . Le pointillé représente le polygone de non-recouvrement entre ces deux polygones.

Le polygone de non-recouvrement de deux polygones P_i et P_j s'écrit mathématiquement de la manière suivante :

$$\begin{aligned} NFP(P_i, P_j) &= \text{int}(P_i) \oplus (-\text{int}(P_j)) \\ &= \{\mathbf{v} - \mathbf{w} \mid \mathbf{v} \in \text{int}(P_i), \mathbf{w} \in \text{int}(P_j)\} \end{aligned} \quad (\text{B.23})$$

où $\text{int}(P)$ désigne l'intérieur du polygone P .

La complexité et les méthodes de calcul des polygones de non-recouvrement de deux polygones dépendent de la géométrie de ces derniers.

B.4.1.1 Cas des polygones convexes

Le calcul des polygones de non-recouvrement dans le cadre de polygones convexes est extrêmement simple et présente une complexité linéaire. La figure B.6 présente les différentes étapes de construction d'un polygone de non-recouvrement de deux polygones convexes. Les segments du premier polygone sont orientés dans le sens anti-horaire, et ceux du deuxième polygone dans le sens horaire. Ensuite, l'ensemble des segments des deux polygones est translaté vers un seul et même point, où il va être ordonné suivant la pente des segments. L'ensemble des vecteurs ordonnés est enfin concaténé pour former le polygone de non-recouvrement. Ce dernier comportera au maximum $m + n$ segments, avec m et n désignant le nombre de segments des polygones.

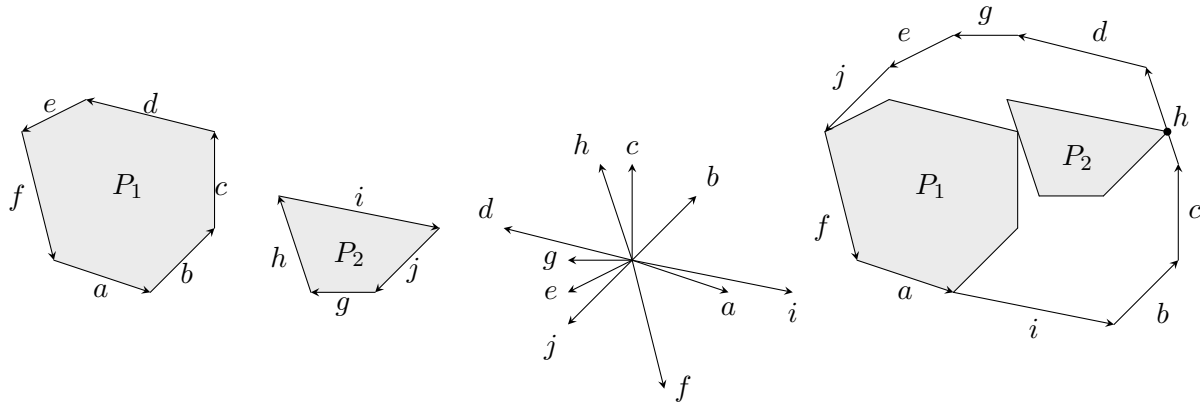


Figure B.6 – Construction du polygone de non-recouvrement pour deux polygones convexes.

B.4.1.2 Cas des polygones quelconques

Avec des polygones non convexes, l'approche présentée précédemment n'est plus valide. La non-convexité des polygones génère une complexité certaine des polygones de non-recouvrement (NFP). En effet, si les polygones P_1 et P_2 ont respectivement m et n segments, alors le nombre de segments de leur NFP sera $O(m^2n^2)$ [AHBN02]. Cette complexité a stimulé les efforts de recherche pour proposer des méthodes de calcul efficaces. Trois approches sont possibles pour calculer les polygones de non-recouvrement :

- La première est basée sur les sommes de Minkowski. C'est en fait une généralisation du cas convexe, où les segments des deux polygones sont triés suivant leurs pentes et leurs précédences [BS08].
- La seconde approche est basée sur les décompositions des polygones en plusieurs polygones convexes. Chaque polygone est décomposé en un ensemble de polygones convexes pour lesquels les polygones de non-recouvrement sont calculés. Le polygone de non-recouvrement final est obtenu en calculant l'union de ces différents polygones. La figure B.7 présente plusieurs des techniques disponibles pour convertir un polygone non convexe en un assemblage de polygones convexes. Agarwal *et al.* [AFH00] ont étudié l'influence des décompositions et recombinaisons sur la rapidité de calcul. Ils ont montré que les décompositions heuristiques sont plus rapides que les décompositions convexes optimales. Ils recommandent une décomposition simultanée des polygones, qui minimise une fonction traduisant le coût de reconstruction du polygone de non-recouvrement.
- Enfin, la troisième approche est basée sur une approche orbitale, où le polygone de non-recouvrement est construit en faisant glisser un polygone autour de l'autre. Basée sur des calculs trigonométriques, cette méthode s'avère extrêmement rapide [BHKW07].

Pour réaliser ces calculs, une approche par décomposition triangulaire a été choisie.

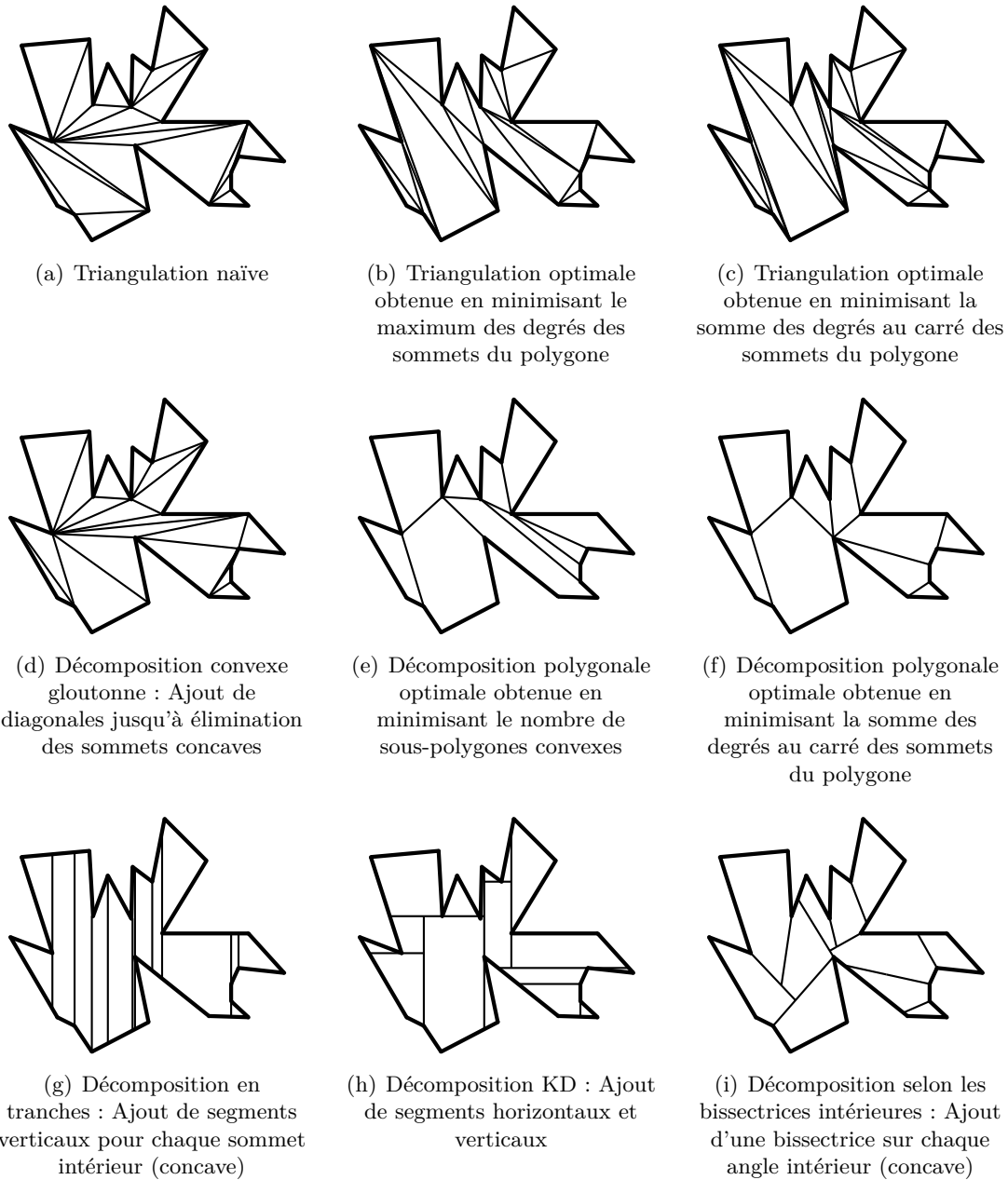


Figure B.7 – Résultats des différentes méthodes de décomposition d'un polygone en un ensemble de sous-polygones convexes. Le degré d'un sommet correspond au nombre de segments qui partent de ce sommet.

B.4.2 Polygones d'appartenance

Le polygone d'appartenance permet de savoir si un polygone P_i est contenu à l'intérieur d'un polygone contenant C . Il est défini mathématiquement de manière suivante

$$IFP(C, P_i) = \overline{NFP(\bar{C}, P_i)} \quad (\text{B.24})$$

avec

$$NFP(\bar{C}, P_i) = \text{int}(\bar{C}) \oplus (-\text{int}(P_i)) = \{v - w \mid v \in \mathbb{R}^2 \setminus C, w \in \text{int}(P_i)\} \quad (\text{B.25})$$

La technique, que nous avons utilisée pour générer ces polygones d'appartenance, consiste à construire une boîte englobante autour du contenant et à considérer le complémentaire du contenant (équivalent de \bar{C}). La figure B.8 illustre ces premières étapes sur un polygone à trou. Ensuite, le polygone de non-recouvrement entre le polygone P_i et le complémentaire est calculé ($NFP(\bar{C}, P_i)$). Le complémentaire de ce résultat donne le polygone d'appartenance : $IFP(C, P_i)$. Dans le cas où le contenant est un rectangle, le polygone d'appartenance sera aussi rectangulaire et peut être calculé directement sans passer par tous ces calculs.

B.4.3 Remarques générales

Développés pour la détection de collision, les sommes de Minkowski, polygones de non-recouvrement et d'appartenance peuvent trouver d'autres applications. En effet, la réalisation de calculs d'accessibilité ou la vérification de contraintes de placement peuvent se baser sur ces outils géométriques. Les polygones d'appartenance peuvent être utilisés pour déterminer si un composant est accessible, ou encore pour générer une trajectoire d'un polygone à l'intérieur d'un polygone plus grand.

Les polygones de non-recouvrement et d'appartenance dépendent de l'orientation des composants. Plusieurs calculs sont donc nécessaires pour prendre en compte les différentes orientations. Toutefois, si le polygone à placer approxime un cercle, un seul calcul est nécessaire pour déterminer l'ensemble des positions que le cercle peut occuper. La figure B.8 montre les différents polygones d'appartenance d'un ensemble de cercles de rayons différents dans un contenant de forme quelconque présentant un trou. Avec des rayons croissants, le polygone d'appartenance forme un ensemble de polygones d'appartenance. Cela signifie que les cercles de rayon 5 et 10 peuvent se déplacer de manière continue dans l'ensemble du contenant, ce qui n'est plus le cas pour les cercles de rayon supérieur. En calculant la somme de Minkowski du polygone d'appartenance avec le polygone à placer, on détermine l'ensemble des points de l'espace qui peut être couvert par le polygone, en l'occurrence un cercle (Figure B.8(i)). Concrètement, le cercle peut modéliser le rayon d'action d'une personne, et le polygone d'appartenance permet de déterminer quels sont les composants accessibles à la personne.

B.5 Angle d'Euler

Les angles d'Euler permettent de représenter l'orientation spatiale d'un composant à l'aide d'une composition de trois rotations. La figure B.9 illustre la rotation d'Euler $z - x - z$ définie par les angles (ϕ, θ, ψ) , qui correspond à une rotation autour de l'axe z , suivie d'une rotation autour du nouvel axe x , et enfin d'une rotation autour du nouvel axe z . L'intersection des plans formés respectivement par les vecteurs (\vec{x}, \vec{y}) et (\vec{X}, \vec{Y}) forme la droite (N) . Sur cette figure :

- ϕ est l'angle formé entre l'axe x et la droite (N) .
- θ est l'angle formé entre l'axe z et l'axe Z .
- ψ est l'angle formé entre la droite (N) et l'axe X .

La matrice rotation totale \mathbf{R} s'écrit donc comme le produit de ces trois rotations élémentaires :

$$[\mathbf{R}] = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{B.26})$$

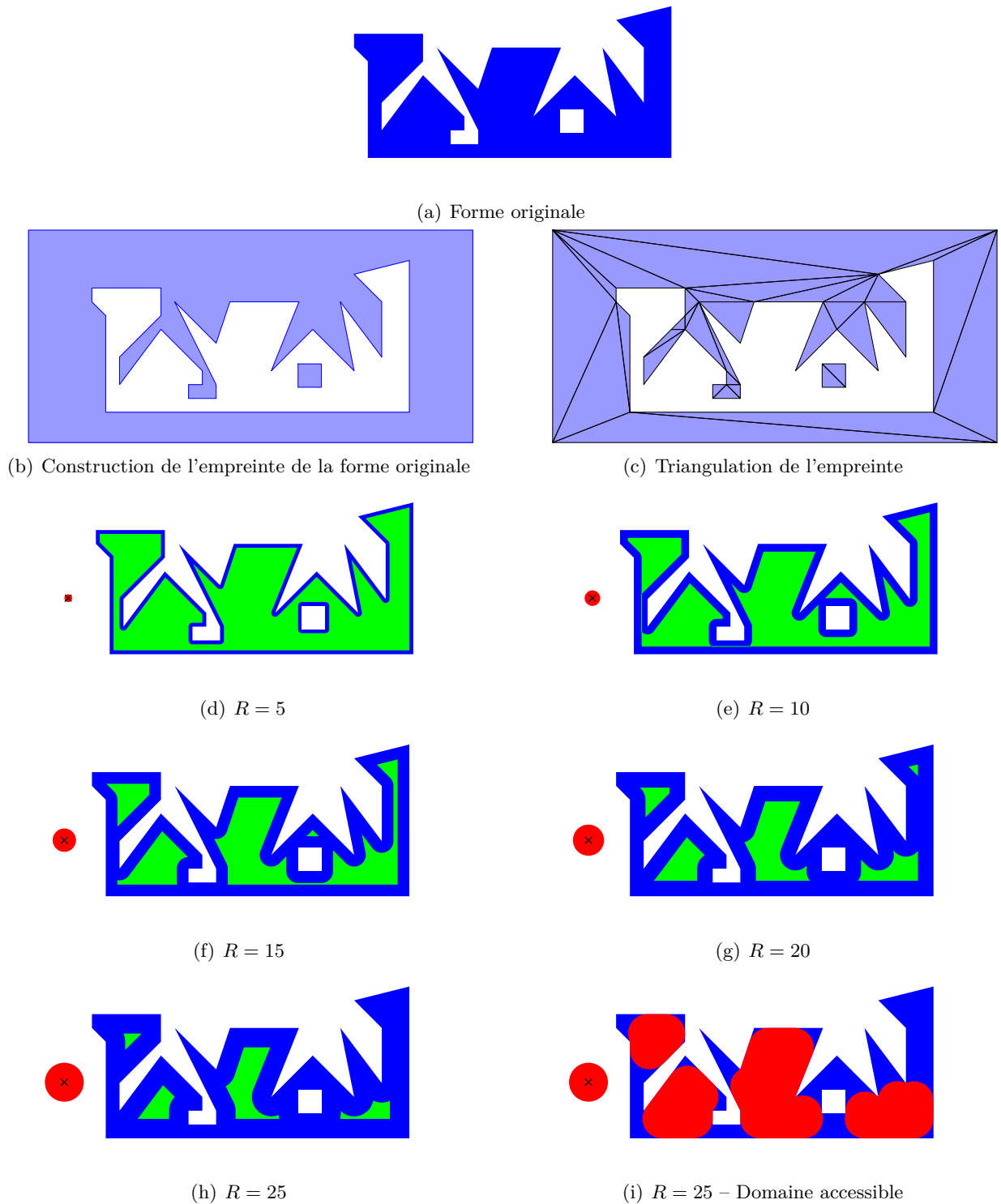


Figure B.8 – Calcul des domaines d'accessibilité 2D à l'aide de calcul des polygones d'appartenance. Les sous-figures (b) et (c) présentent les prétraitements effectués pour obtenir le polygone d'appartenance d'un polygone avec ce contenant. Les sous-figures suivantes (d à h) donnent les polygones d'appartenance pour des cercles de différents rayons. Si les centres des cercles appartiennent à ces polygones, alors ils sont entièrement contenus dans le contenant. La sous-figure (i) présente le domaine d'accessibilité pour un cercle de rayon 25. Ce domaine est obtenu en calculant la somme de Minkowski du polygone d'appartenance avec le cercle de rayon 25.

Une fois les produits de matrice effectués, la matrice de rotation \mathbf{R} s'écrit :

$$[\mathbf{R}] = \begin{bmatrix} \cos \phi \cos \psi - \cos \theta \sin \phi \sin \psi & -\cos \psi \sin \phi - \cos \phi \cos \theta \sin \psi & \sin \theta \sin \psi \\ \cos \theta \cos \psi \sin \phi + \cos \phi \sin \psi & \cos \phi \cos \theta \cos \psi - \sin \phi \sin \psi & -\cos \psi \sin \theta \\ \sin \phi \sin \theta & \cos \phi \sin \theta & \cos \theta \end{bmatrix} \quad (\text{B.27})$$

Les dérivées partielles de la matrice de rotation \mathbf{R} s'écrivent :

$$\begin{aligned} \frac{\partial \mathbf{R}}{\partial \phi} &= \begin{bmatrix} -\sin \phi \cos \psi - \cos \theta \cos \phi \sin \psi & -\cos \psi \cos \phi + \sin \phi \cos \theta \sin \psi & 0 \\ \cos \theta \cos \psi \cos \phi - \sin \phi \sin \psi & -\sin \phi \cos \theta \cos \psi - \cos \phi \sin \psi & 0 \\ \cos \phi \sin \theta & -\sin \phi \sin \theta & 0 \end{bmatrix} \\ \frac{\partial \mathbf{R}}{\partial \theta} &= \begin{bmatrix} \sin \theta \sin \phi \sin \psi & \cos \phi \sin \theta \sin \psi & \cos \theta \sin \psi \\ -\sin \theta \cos \psi \sin \phi & -\cos \phi \sin \theta \cos \psi & -\cos \psi \cos \theta \\ \sin \phi \cos \theta & \cos \phi \cos \theta & -\sin \theta \end{bmatrix} \\ \frac{\partial \mathbf{R}}{\partial \psi} &= \begin{bmatrix} -\cos \phi \sin \psi - \cos \theta \sin \phi \cos \psi & \sin \psi \sin \phi - \cos \phi \cos \theta \cos \psi & \sin \theta \cos \psi \\ -\cos \theta \sin \psi \sin \phi + \cos \phi \cos \psi & -\cos \phi \cos \theta \sin \psi - \sin \phi \cos \psi & \sin \psi \sin \theta \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \quad (\text{B.28})$$

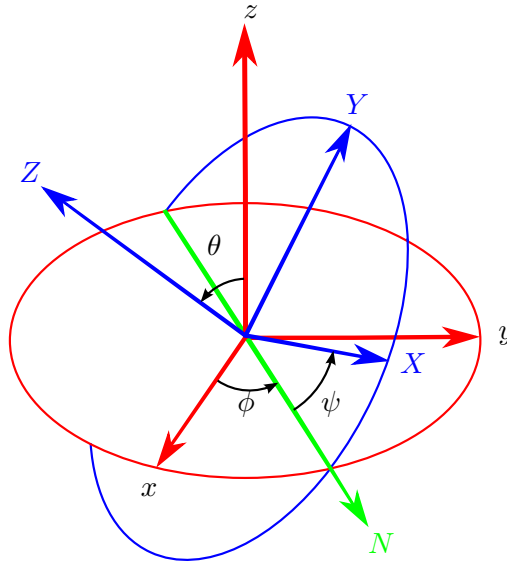


Figure B.9 – Angle d'Euler $z - x - z$.

Annexe C

Résultats annexes

Ce chapitre regroupe les résultats numériques du chapitre 5 qui n'ont pu être présentés par souci de lisibilité. La structure de ce chapitre est identique à celle du chapitre de résultats.

C.1 Résultats annexes obtenus pour un 1^{er} problème d'agencement

Cette section contient les résultats numériques associés aux quatre simulations \mathcal{S}_1 , \mathcal{S}_2 , \mathcal{S}_3 et \mathcal{S}_4 , présentées section 5.2. Les simulations \mathcal{S}_1 et \mathcal{S}_2 font varier la probabilité de mutation des variables réelles p_m avec deux probabilités d'échange de 0% et 30%. Les simulations \mathcal{S}_3 et \mathcal{S}_4 font varier la probabilité d'échange p_s avec respectivement $p_m = 0\%$ et $p_m = 30\%$.

C.1.1 Rôle des opérateurs génétiques

C.1.1.1 Comparaison des hypervolumes relatifs

Les tables C.1, C.2, C.3 et C.4 présentent les résultats statistiques de l'hypervolume relatif pour les quatre simulations \mathcal{S}_1 , \mathcal{S}_2 , \mathcal{S}_3 et \mathcal{S}_4 .

p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
0%	0.0717	0.1500	0.2745	0.0779	0.0773	0.1205	0.2355
10%	0.6676	0.8106	0.8930	0.0728	0.7908	0.8238	0.8645
20%	0.6575	0.8147	0.9041	0.0771	0.8004	0.8349	0.8566
30%	0.7778	0.8885	0.9597	0.0593	0.8501	0.9085	0.9265
40%	0.7837	0.8651	0.9521	0.0559	0.8227	0.8650	0.9022
50%	0.8215	0.8585	0.9418	0.0388	0.8427	0.8434	0.8583
60%	0.7388	0.8205	0.9041	0.0504	0.7944	0.8199	0.8462
70%	0.7225	0.8312	0.9175	0.0796	0.7320	0.8571	0.9013
80%	0.7418	0.8169	0.8954	0.0462	0.7894	0.8214	0.8324
90%	0.7230	0.7783	0.8693	0.0522	0.7306	0.7669	0.8134
100%	0.7012	0.7847	0.8293	0.0432	0.7692	0.7916	0.8255

Table C.1 – Statistiques sur l'hypervolume relatif pour \mathcal{S}_1 (Évolution de la probabilité de mutation avec $p_s = 0\%$).

p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
0%	0.4377	0.6602	0.7195	0.1010	0.6696	0.7077	0.7191
10%	0.7775	0.8332	0.8819	0.0355	0.8078	0.8308	0.8707
20%	0.8302	0.8649	0.9025	0.0281	0.8356	0.8671	0.8870
30%	0.8141	0.8752	0.9076	0.0340	0.8503	0.8864	0.9064
40%	0.7644	0.8340	0.9395	0.0566	0.7908	0.8277	0.8539
50%	0.8029	0.8465	0.9182	0.0395	0.8098	0.8369	0.8747
60%	0.6404	0.7780	0.9125	0.0890	0.7191	0.7772	0.8417
70%	0.7462	0.8180	0.8671	0.0379	0.7996	0.8275	0.8398
80%	0.7842	0.8167	0.9004	0.0391	0.7869	0.8065	0.8154
90%	0.7621	0.7734	0.7895	0.0097	0.7629	0.7733	0.7794
100%	0.6998	0.7603	0.7980	0.0322	0.7392	0.7727	0.7794

Table C.2 – Statistiques sur l’hypervolume relatif pour \mathcal{S}_2 (Évolution de la probabilité de mutation avec $p_s = 30\%$).

p_s	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
0%	0.0717	0.1500	0.2745	0.0779	0.0773	0.1205	0.2355
10%	0.3627	0.6148	0.7434	0.1206	0.6049	0.6492	0.6796
20%	0.3813	0.6270	0.7364	0.1296	0.5395	0.6846	0.7357
30%	0.4521	0.6757	0.7747	0.1097	0.6307	0.7213	0.7540
40%	0.4669	0.7084	0.8377	0.1154	0.7161	0.7327	0.7642
50%	0.4855	0.6830	0.7721	0.0956	0.6728	0.7035	0.7607
60%	0.5557	0.7145	0.8097	0.0828	0.6933	0.7213	0.7857
70%	0.6506	0.7283	0.8192	0.0602	0.6786	0.7168	0.7876
80%	0.6567	0.7225	0.7693	0.0343	0.7103	0.7307	0.7372
90%	0.6890	0.7564	0.8180	0.0400	0.7412	0.7518	0.7866
100%	0.7170	0.7667	0.8187	0.0343	0.7321	0.7726	0.7869

Table C.3 – Statistiques sur l’hypervolume relatif pour \mathcal{S}_3 (Évolution de la probabilité d’échange avec $p_m = 0\%$).

p_s	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
0%	0.7778	0.8885	0.9597	0.0593	0.8501	0.9085	0.9265
10%	0.7663	0.8582	0.9378	0.0580	0.8008	0.8727	0.8990
20%	0.8007	0.8517	0.9591	0.0524	0.8185	0.8312	0.8696
30%	0.8449	0.8736	0.9093	0.0261	0.8477	0.8672	0.9057
40%	0.7792	0.8602	0.9269	0.0466	0.8459	0.8543	0.9003
50%	0.7998	0.8634	0.9002	0.0427	0.8082	0.8866	0.8992
60%	0.7714	0.8405	0.8928	0.0376	0.8314	0.8397	0.8680
70%	0.7428	0.8175	0.8713	0.0388	0.8061	0.8247	0.8355
80%	0.7956	0.8437	0.8950	0.0361	0.8143	0.8367	0.8840
90%	0.7752	0.8091	0.8371	0.0207	0.7932	0.8128	0.8237
100%	0.7744	0.8179	0.8659	0.0381	0.7814	0.8118	0.8622

Table C.4 – Statistiques sur l’hypervolume relatif pour \mathcal{S}_4 (Évolution de la probabilité d’échange avec $p_m = 30\%$).

C.1.1.2 Comparaison des surfaces de compromis à l'aide de la métrique \mathcal{C}

Les tables C.5, C.6, C.7 et C.8 présentent des comparaisons des surfaces de compromis pour les quatre simulations \mathcal{S}_1 , \mathcal{S}_2 , \mathcal{S}_3 et \mathcal{S}_4 à l'aide de la métrique \mathcal{C} , dont la définition est donnée en section 2.4.2.1. Le niveau de gris des cellules met en évidence le niveau de dominance d'une surface de compromis par rapport à une autre.

	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
0%	0	0	0.0093	0.0120	0	0	0	0	0	0	0
10%	1	0	0.5888	0.0120	0	0.0667	0.1538	0	0.9167	1	1
20%	1	0.2473	0	0.0120	0	0.0667	0.1538	0	0.7500	1	0.9143
30%	1	1	1	0	0.8000	0.9667	1	1	1	1	1
40%	1	1	1	0.0964	0	0.8333	0.9615	0.8696	1	1	1
50%	1	0.9140	0.9720	0.0120	0.1250	0	0.6923	0.7391	0.9722	1	1
60%	1	0.9032	0.8411	0.0120	0	0.3333	0	0.2174	0.9444	0.9655	1
70%	1	1	0.9626	0.0120	0.0250	0.2667	0.5769	0	0.9722	1	1
80%	1	0.0968	0.0280	0.0120	0	0.0667	0.1154	0	0	0.5862	0.8857
90%	1	0	0.0093	0.0120	0	0	0.1154	0	0.2500	0	0.4286
100%	1	0	0.0187	0.0120	0	0	0	0	0.0556	0.4483	0

Table C.5 – Comparaison des surfaces de compromis de \mathcal{S}_1 à l'aide de la métrique \mathcal{C} . (Évolution de la probabilité de mutation avec $p_s = 0\%$).

	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
0%	0	0.0172	0	0	0	0	0	0	0	0.0345	0.0690
10%	1	0	0.1681	0.2436	0	0.0526	0.0833	0.8000	0.1111	1	1
20%	1	0.7845	0	0.6667	0	0.0789	0.0833	0.9714	0.1111	1	1
30%	1	0.6638	0.3277	0	0	0.0789	0.0833	0.8286	0.1111	1	1
40%	1	0.9138	0.8655	0.9615	0	0.8947	1	1	1	1	1
50%	1	0.8707	0.8403	0.8974	0.0526	0	0.5833	1	0.6111	1	1
60%	1	0.8276	0.8235	0.8590	0	0.3421	0	0.9714	0.7778	1	1
70%	1	0.2500	0.0168	0.1154	0	0	0.0833	0	0.1111	0.9655	0.9655
80%	1	0.8017	0.8235	0.8205	0	0.1842	0.1250	0.9714	0	1	1
90%	0.9149	0.0172	0	0	0	0	0	0.0286	0	0	0.3103
100%	0.9574	0.0172	0	0	0	0	0	0	0	0.5172	0

Table C.6 – Comparaison des surfaces de compromis de \mathcal{S}_2 à l'aide de la métrique \mathcal{C} . (Évolution de la probabilité de mutation avec $p_s = 30\%$)

	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
0%	0	0	0	0	0	0	0	0	0	0	0
10%	1	0	0.1481	0.3125	0	0.1176	0.1034	0	0.3800	0	0
20%	1	0.3333	0	0.5000	0.0345	0.1176	0	0.2857	0.3000	0	0.0600
30%	1	0.6667	0.3333	0	0	0.1176	0.1724	0.2381	0.3000	0	0.1000
40%	1	0.9444	0.8148	0.7500	0	0.4118	0.3448	0.5238	0.5400	0.2609	0.3600
50%	1	0.4444	0.7778	0.6250	0.4483	0	0.2414	0.7143	0.7800	0.5217	0.5800
60%	1	0.9444	1	0.8750	0.4828	0.5294	0	0.8095	0.8800	0.7391	0.7800
70%	1	0.8889	0.1852	0.4375	0.0690	0.2353	0.1034	0	0.7400	0.0870	0.2800
80%	1	0.1667	0.1111	0.3750	0.1034	0.0588	0.1034	0.1905	0	0	0.1000
90%	1	0.9444	0.9630	0.8750	0.2414	0.4118	0.2069	0.5714	0.5800	0	0.3000
100%	1	0.9444	0.6296	0.9375	0.4138	0.3529	0.1379	0.4286	0.6200	0.5217	0

Table C.7 – Comparaison des surfaces de compromis de \mathcal{S}_3 à l'aide de la métrique \mathcal{C} . (Évolution de la probabilité d'échange avec $p_m = 0\%$)

	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
0%	0	0.8507	0.8889	1	0.9545	1	1	1	1	1	1
10%	0.0241	0	0.3333	0.9315	0.5606	0.9595	0.9048	0.9818	0.9444	1	1
20%	0.0241	0.5821	0	0.9589	0.6818	0.9459	0.9048	1	0.9815	1	1
30%	0.0120	0.0448	0.0185	0	0.0152	0.4865	0.2381	0.8545	0.8333	0.9677	0.4091
40%	0.0120	0.1343	0.2037	0.9315	0	0.9324	1	1	1	1	1
50%	0.0120	0.0448	0.0185	0.5068	0.0152	0	0.1429	0.9636	0.8889	0.9839	0.4091
60%	0.0120	0.0448	0.0185	0.6712	0.0152	0.6216	0	0.8909	0.7222	1	0.7727
70%	0.0120	0.0448	0.0185	0.0411	0.0152	0.0135	0.0952	0	0.1481	0.8387	0.3182
80%	0.0120	0.0448	0.0185	0.1370	0.0152	0	0.0476	0.8545	0	0.9355	0.3636
90%	0.0120	0.0149	0.0185	0.0411	0.0152	0	0	0.1273	0.0370	0	0.1364
100%	0.0120	0.0149	0.0185	0.6301	0.0152	0.5405	0.0952	0.8545	0.5000	0.8065	0

Table C.8 – Comparaison des surfaces de compromis de \mathcal{S}_4 à l’aide de la métrique \mathcal{C} . (Évolution de la probabilité d’échange avec $p_m = 30\%$)

C.1.2 Choix de l’algorithme d’optimisation globale

C.1.2.1 Comparaison sur l’hypervolume relatif

La table C.9 présente les statistiques sur l’hypervolume relatif pour les algorithmes *NSGA-II* et *Omni-Optimizer*. Ces statistiques sont présentées graphiquement sur la figure 5.23. Les résultats donnent l’avantage à l’algorithme *Omni-Optimizer*.

Algorithme	p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
<i>NSGA-II</i>	10%	0.6511	0.8063	0.9404	0.1051	0.6837	0.8313	0.9146
<i>Omni-Optimizer</i>	10%	0.6719	0.8106	0.9458	0.0889	0.7072	0.8533	0.8749
<i>NSGA-II</i>	20%	0.6067	0.8012	0.9411	0.1143	0.6793	0.8389	0.8832
<i>Omni-Optimizer</i>	20%	0.6176	0.8196	0.9693	0.0969	0.7572	0.8453	0.8911
<i>NSGA-II</i>	30%	0.6266	0.8388	0.9419	0.0870	0.8275	0.8613	0.9002
<i>Omni-Optimizer</i>	30%	0.7046	0.8554	0.9627	0.0572	0.8095	0.8658	0.9008
<i>NSGA-II</i>	40%	0.6415	0.8051	0.8860	0.0756	0.7484	0.8381	0.8652
<i>Omni-Optimizer</i>	40%	0.6315	0.8381	0.9696	0.0855	0.7836	0.8568	0.9045
<i>NSGA-II</i>	50%	0.6578	0.8380	0.9451	0.0682	0.7920	0.8500	0.8875
<i>Omni-Optimizer</i>	50%	0.6967	0.8542	0.9467	0.0563	0.8395	0.8496	0.8999
<i>NSGA-II</i>	60%	0.5912	0.8188	0.9289	0.0859	0.7724	0.8446	0.8887
<i>Omni-Optimizer</i>	60%	0.5958	0.8307	0.9235	0.0787	0.8064	0.8418	0.8879

Table C.9 – Statistiques sur l’hypervolume relatif pour les algorithmes *NSGA-II* et *Omni-Optimizer*. (Évolution de la probabilité de mutation avec $p_s = 10\%$).

C.1.2.2 Comparaison sur la diversité génotypique

Les tables C.10 et C.11 présentent les statistiques sur les distances génotypiques des solutions de surfaces de compromis. Ces statistiques sont représentées graphiquement figure 5.25. Ces résultats montrent que l'algorithme *Omni-Optimizer* conserve davantage la diversité génotypique que l'algorithme *NSGA-II*. Plus ces distances sont importantes, meilleure est la diversité génotypique des solutions.

<i>Algorithme</i>	p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
<i>NSGA-II</i>	10%	0.1103	0.1754	0.2886	0.0502	0.1337	0.1637	0.2180
<i>Omni-Optimizer</i>	10%	0.1606	0.2451	0.3329	0.0488	0.2035	0.2455	0.2816
<i>NSGA-II</i>	20%	0.0929	0.1661	0.2825	0.0529	0.1304	0.1494	0.2075
<i>Omni-Optimizer</i>	20%	0.1506	0.2475	0.3376	0.0548	0.2016	0.2572	0.2863
<i>NSGA-II</i>	30%	0.1293	0.1936	0.2580	0.0382	0.1585	0.1956	0.2221
<i>Omni-Optimizer</i>	30%	0.1318	0.2195	0.3514	0.0533	0.1844	0.2138	0.2478
<i>NSGA-II</i>	40%	0.1389	0.2110	0.3011	0.0477	0.1779	0.2082	0.2511
<i>Omni-Optimizer</i>	40%	0.1106	0.2178	0.2846	0.0504	0.1911	0.2312	0.2610
<i>NSGA-II</i>	50%	0.1263	0.2116	0.3704	0.0586	0.1710	0.1891	0.2410
<i>Omni-Optimizer</i>	50%	0.1827	0.2414	0.3333	0.0366	0.2143	0.2456	0.2552
<i>NSGA-II</i>	60%	0.1126	0.2182	0.3095	0.0478	0.1783	0.2215	0.2531
<i>Omni-Optimizer</i>	60%	0.1592	0.2425	0.3724	0.0567	0.2056	0.2249	0.2655

Table C.10 – Mesures statistiques sur les distances génotypiques moyennes pour les algorithmes *NSGA-II* et *Omni-Optimizer*. (Évolution de la probabilité de mutation avec $p_s = 10\%$).

<i>Algorithme</i>	p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
<i>NSGA-II</i>	10%	0.2467	0.4222	0.5260	0.0756	0.3863	0.4363	0.4823
<i>Omni-Optimizer</i>	10%	0.3411	0.4694	0.5785	0.0589	0.4250	0.4825	0.5109
<i>NSGA-II</i>	20%	0.2231	0.3785	0.5281	0.0785	0.3152	0.3954	0.4376
<i>Omni-Optimizer</i>	20%	0.3495	0.4900	0.5785	0.0563	0.4590	0.4847	0.5381
<i>NSGA-II</i>	30%	0.2796	0.4307	0.5253	0.0739	0.3960	0.4347	0.4966
<i>Omni-Optimizer</i>	30%	0.2681	0.4543	0.5721	0.0684	0.4115	0.4529	0.5067
<i>NSGA-II</i>	40%	0.2850	0.4350	0.5883	0.0712	0.4095	0.4347	0.4789
<i>Omni-Optimizer</i>	40%	0.2189	0.4344	0.5463	0.0849	0.3934	0.4494	0.5061
<i>NSGA-II</i>	50%	0.2837	0.4123	0.5479	0.0870	0.3189	0.4053	0.4879
<i>Omni-Optimizer</i>	50%	0.3420	0.4923	0.5956	0.0606	0.4574	0.4969	0.5425
<i>NSGA-II</i>	60%	0.2524	0.4075	0.5470	0.0677	0.3741	0.4081	0.4274
<i>Omni-Optimizer</i>	60%	0.2525	0.4666	0.6068	0.0999	0.4206	0.4734	0.5437

Table C.11 – Mesures statistiques sur les distances génotypiques maximales pour les algorithmes *NSGA-II* et *Omni-Optimizer*. (Évolution de la probabilité de mutation avec $p_s = 10\%$).

C.2 Résultats annexes obtenus pour un 2^{ème} problème d'agencement

C.2.1 Réglages des opérateurs génétiques

C.2.1.1 Comparaison des hypervolumes relatifs

Les tables C.12 et C.13 présentent numériquement les statistiques concernant l'hypervolume relatif pour les variations des probabilités des opérateurs de mutation et d'échange. Plus les résultats sont proches de 1, meilleure est l'approximation du front de Pareto lors des différentes optimisations.

p_m	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
0%	0	0.0334	0.3825	0.0868	0	0	0
10%	0	0.3081	0.7430	0.2334	0.0451	0.3495	0.4839
20%	0	0.3595	0.9559	0.2776	0.0916	0.3854	0.6013
30%	0	0.2707	0.7095	0.2216	0.0321	0.2336	0.4304
40%	0	0.1891	0.6626	0.2025	0	0.1284	0.3188
50%	0	0.1948	0.6769	0.1753	0.0052	0.1832	0.3176
60%	0	0.1818	0.6536	0.1831	0.0031	0.1406	0.2948
70%	0	0.0912	0.5472	0.1108	0	0.0534	0.1496
80%	0	0.0897	0.4263	0.1116	0	0.0503	0.1534
90%	0	0.0555	0.3900	0.0852	0	0.0002	0.0749
100%	0	0.0617	0.4004	0.1007	0	0	0.0868

Table C.12 – Statistiques sur l'hypervolume relatif en fonction de la probabilité de mutation p_m .

p_s	min	moy.	max	σ	1 ^{er} quart.	méd.	3 ^{ème} quart.
0%	0	0.2780	0.7183	0.2346	0.0277	0.2485	0.4640
10%	0	0.2818	0.7532	0.2496	0.0059	0.2431	0.5060
20%	0	0.2825	0.6927	0.2171	0.0729	0.2205	0.4895
30%	0	0.2707	0.7095	0.2216	0.0321	0.2336	0.4304
40%	0	0.2880	0.9225	0.2005	0.1130	0.2815	0.4054
50%	0	0.3470	0.8623	0.2224	0.2395	0.3574	0.4983
60%	0	0.3394	0.7456	0.2181	0.1517	0.3328	0.4897
70%	0	0.4234	0.8466	0.2172	0.3132	0.4197	0.6162
80%	0	0.4280	0.7540	0.2156	0.3149	0.4731	0.5982
90%	0	0.3380	0.8612	0.2011	0.1723	0.3870	0.4957
100%	0	0.2724	0.8078	0.1713	0.1379	0.2830	0.3350

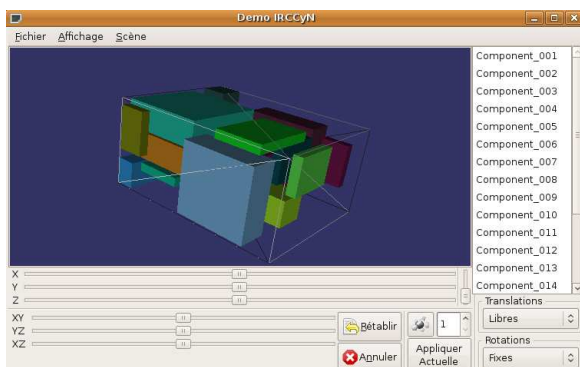
Table C.13 – Statistiques sur l'hypervolume relatif en fonction de la probabilité d'échange p_s .

Annexe D

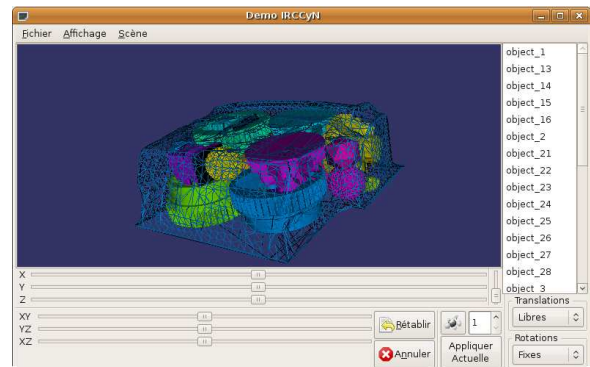
Présentation du démonstrateur développé

Parallèlement au déroulement de cette thèse, l'*IRCCyN* a financé le développement d'un environnement 3D pour la résolution de problèmes d'agencement.

Basé sur la plateforme 3D *OpenSceneGraph*¹, le toolkit *wxWidgets*², et la bibliothèque 3D *GTS*³, ce démonstrateur a pour objectif de tester les différentes méthodes de résolution proposées. Il présente des fonctionnalités graphiques, d'interaction et de résolution. Il permet actuellement de générer les données du problème à traiter, de visualiser des scènes 3D à partir de tout type de fichiers 3D, de déplacer manuellement ou numériquement l'ensemble des composants pour la prise en compte future de l'interaction entre l'utilisateur. Le démonstrateur utilise l'algorithme génétique *Omni-Optimizer* comme optimiseur global, ainsi que l'algorithme *BFGS* pour réaliser les séparations. Actuellement, seul l'algorithme de séparation pour parallélépipèdes est implémenté. La représentation des composants en assemblages de sphères est déjà implémentée. L'intégration de l'algorithme de séparation 3D pour composants quelconques va permettre la résolution des problèmes 3D avec des géométries complexes. La figure D.1 présente deux captures d'écran du démonstrateur développé. La sous-figure (a) présente une solution efficace pour la résolution d'un problème d'agencement de parallélépipèdes. La sous-figure (b) montre l'intégration des résultats de *packing 3D* issus des programmes de Tiwari *et al.* [TFF08].



(a) Résolution de problème d'agencement dans le démonstrateur



(b) Intégration des résultats de packing dans le démonstrateur

Figure D.1 – Captures d'écran du démonstrateur développé.

La figure D.2 présente l'architecture modulaire du logiciel développé. Cette architecture est développée de manière à intégrer de nouveaux composants, ainsi que les interactions entre le concepteur et les algorithmes d'optimisation, qui nous semblent essentielles.

1. www.openscenegraph.org
2. www.wxwidgets.org/
3. <http://gts.sourceforge.net/>

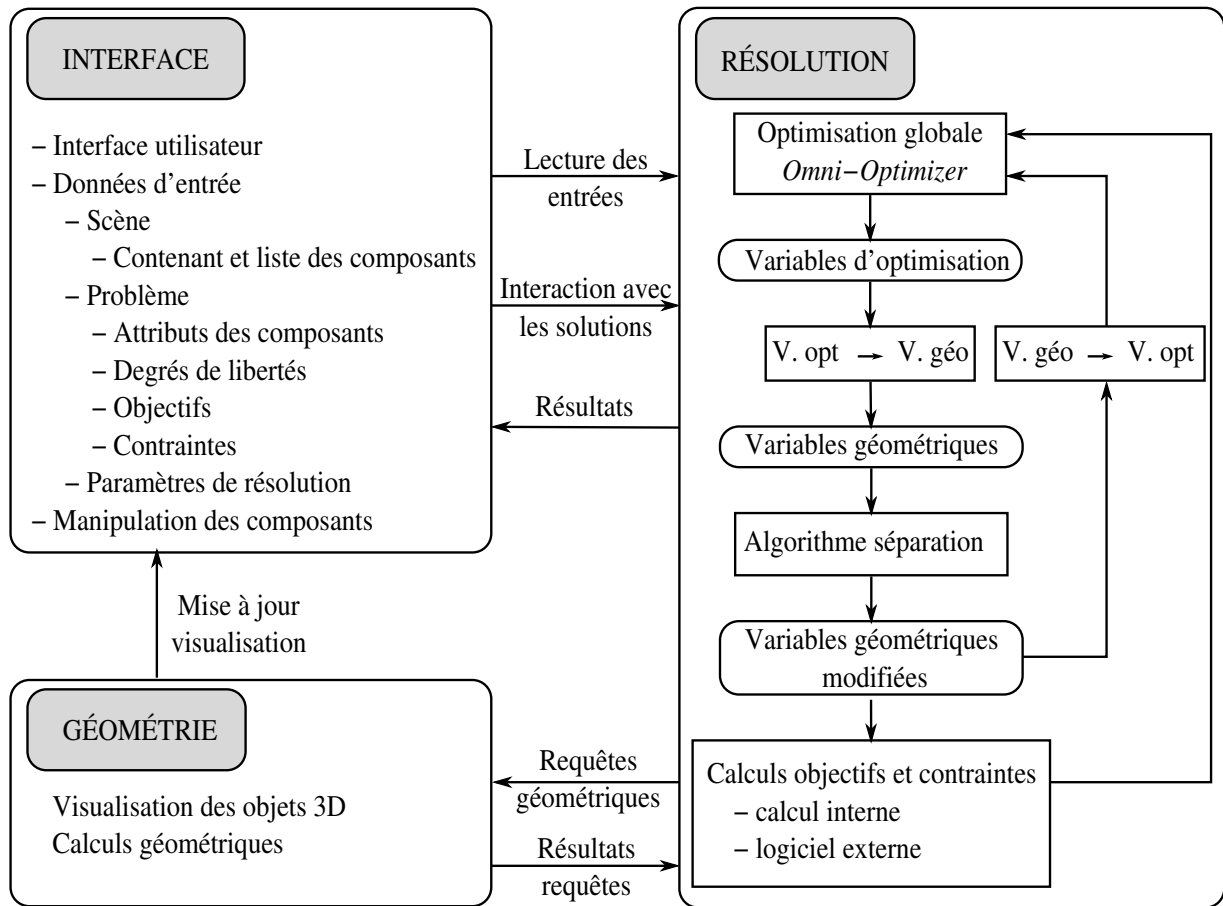


Figure D.2 – Architecture du démonstrateur développé

En vue de l'industrialisation de la méthode proposée, plusieurs développements techniques sont nécessaires. Il faut :

- terminer l'intégration de l'algorithme de séparation dans le démonstrateur ;
- développer une interface pour faciliter la mise en données des problèmes ;
- travailler sur un langage de modélisation des contraintes et des objectifs pour éviter à l'utilisateur d'avoir à entrer dans le code ;
- intégrer des outils de post-traitements : afficher les données numériques des objectifs et contraintes sur la scène, visualiser le front de Pareto, analyser la convergence des optimisations, comparer les résultats de deux optimisations, ... ;
- développer une communication avec des logiciels externes pour le calcul d'objectifs spécifiques.

Bibliographie

- [AB63] Gordon C. ARMOUR et Elwood S. BUFFA : A heuristic algorithm and simulation approach to relative location of facilities. *International Journal of Management Science*, 9(1): 294–309, 1963.
1 citation sur la page 44.
- [ABHI05] Abdul-Rahim AHMAD, Otman BASIR, Khaled HASSANEIN et Muhammad Hasan IMAM : A hierarchical placement strategy for generating superior layout decision alternatives. *International Journal of Quantitative Management*, 11(4): 1–20, 2005.
3 citations sur la page 142.
- [ACS07a] Chandankumar ALADAHALLI, Jonathan CAGAN et Kenji SHIMADA : Objective function effect based pattern search - An implementation for 3D component layout. *Journal of Mechanical Design*, 129(3): 255–265, mars 2007.
4 citations sur les pages 22, 138 et 141.
- [ACS07b] Chandankumar ALADAHALLI, Jonathan CAGAN et Kenji SHIMADA : Objective function effect based pattern search - Theoretical framework inspired by 3D component layout. *Journal of Mechanical Design*, 129(3): 243–254, mars 2007.
2 citations sur les pages 22 et 141.
- [AdC08] Cláudio ALVES et J. M. Valério de CARVALHO : A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, 35(4): 1315–1328, 2008.
1 citation sur la page 141.
- [AEG06] Giuseppe AIELLO, Mario ENEA et Giacomo GALANTE : A multi-objective approach to facility layout problem by genetic search algorithm and Electre method. *Robotics and Computer-Integrated Manufacturing*, 22: 447–455, 2006.
1 citation sur la page 18.
- [AFH00] Pankaj K. AGARWAL, Eyal FLATO et Dan HALPERIN : Polygon decomposition for efficient construction of Minkowski sums. Dans *ESA '00: Proceedings of the 8th Annual European Symposium on Algorithms*, pages 20–31, London, UK, 2000. Springer-Verlag.
1 citation sur la page 157.
- [AGHP⁺00] Pankaj K. AGARWAL, Leonidas J. GUIBAS, Sariel HAR-PELED, Alexander RABINOVITCH et Micha SHARIR : Penetration depth of two convex polytopes in 3D. *Nordic Journal of Computing*, 7(3): 227–240, 2000.
1 citation sur la page 83.
- [AHBN02] Tetsuo ASANO, Antonio HERNÁNDEZ-BARRERA et Subhas C. NANDY : Translating a convex polyhedron over monotone polyhedra. *Computational Geometry: Theory and Applications*, 23(3): 257–269, 2002.
1 citation sur la page 157.
- [AS80] Antonio ALBANO et Guisepppe SAPUPPO : Optimal allocation of two-dimensional irregular shapes using heuristic search methods. *IEEE Transactions on Systems, Man and Cybernetics*, 5: 242–248, 1980.
1 citation sur la page 102.

- [Att95] Dominique ATTALI : *Squelettes et graphes de Voronoï 2D et 3D*. Thèse de doctorat, Institut d'Informatique et de Mathématiques Appliquées de Grenoble, 1995.
1 citation sur la page 65.
- [BCP08] Nicolas BELDICEANU, Mats CARLSSON et Emmanuel PODER : New filtering for the cumulative constraint in the context of non-overlapping rectangles. Dans *CPAIOR*, éditeurs : Laurent PERRON et Michael A. TRICK, volume 5015 de *Lecture Notes in Computer Science*, pages 21–35. Springer, 2008.
1 citation sur la page 27.
- [BDD01] Julia A. BENNELL, Kathryn A. DOWSLAND et William. B. DOWSLAND : The irregular cutting-stock problem – a new procedure for deriving the no-fit polygon. *Computers & Operations Research*, 28(3): 271–287, mars 2001.
2 citations sur la page 142.
- [Ben78] Harold P. BENSON : Existence of efficient solutions for vector maximization problems. *Journal of Optimization Theory and Applications*, 26(4): 569–580, 1978.
1 citation sur la page 31.
- [Ben82] Bengt-Erik BENGTTSSON : Packing rectangular pieces – a heuristic approach. *The Computer Journal*, 25: 353–357, 1982.
1 citation sur la page 14.
- [BHKW07] Edmund K. BURKE, Robert S.R. HELLIER, Graham KENDALL et Glenn WHITWELL : Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, 127(1): 27–49, juillet 2007.
2 citations sur les pages 155 et 157.
- [BHW93] Jacek BŁAŻEWICZ, Piotr HAWRYLUK et Rafała WALKOWIAK : Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Annals of Operations Research*, 41(1-4): 313–325, 1993.
1 citation sur la page 67.
- [BKS96] Richard H. BYRD, Humaid Fayez KHALFAN et Robert B. SCHNABEL : Analysis of a symmetric rank-one trust region method. *SIAM Journal on Optimization*, 6(4): 1025–1039, 1996.
1 citation sur la page 147.
- [BKW04] Edmund K. BURKE, Graham KENDALL et Glenn WHITWELL : A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4): 655–671, 2004.
1 citation sur la page 12.
- [BLNZ95] Richard H. BYRD, Peihuang LU, Jorge NOCEDAL et Ciyu ZHU : A limited memory algorithm for bound constrained optimization. *SIAM – Journal on Scientific Computing*, 16(5): 1190–1208, 1995.
2 citations sur les pages 69 et 147.
- [BM90] Eberhard E. BISCHOFF et Michael D. MARRIOTT : A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, 44: 266–276, 1990.
1 citation sur la page 13.
- [BME94] Yavuz A. BOZER, Russell D. MELLER et Steven J. ERLEBACHER : An improved type layout algorithm for single and multi-floor facilities. *International Journal of Management Science*, 40: 918–950, 1994.
1 citation sur la page 44.
- [BMN05] Ernesto G. BIRGIN, Reinaldo MORÁBITO et Fabio H. NISHIHARA : A note on an L-approach for solving the manufacturer's pallet loading problem. *Journal of the Operational Research Society*, 56: 1448–1451, 2005.
1 citation sur la page 15.

- [BNZ97] Richard H. BYRD, Jorge NOCEDAL et Ciyou ZHU : L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4): 550–560, 1997.
2 citations sur les pages 69 et 147.
- [BO02] Gareth BRADSHAW et Carol O’SULLIVAN : Sphere-tree construction using dynamic medial axis approximation. Dans *SCA ’02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 33–40, New York, NY, USA, 2002. ACM.
1 citation sur la page 94.
- [BO04] Gareth BRADSHAW et Carol O’SULLIVAN : Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics*, 23: 1–26, 2004.
2 citations sur les pages 94 et 95.
- [Box57] George Edward Pelham BOX : Evolutionary operation: a method for increasing industrial productivity. *Journal of Applied Statistics*, 6(2): 81–101, 1957.
1 citation sur la page 22.
- [Bro70] Charles G. BROYDEN : The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6: 76–90, 1970.
2 citations sur les pages 62 et 147.
- [BS07] Julia A. BENNELL et Xiang SONG : A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, 2007.
5 citations sur les pages 4, 15, 102 et 141.
- [BS08] Julia A. BENNELL et Xiang SONG : A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums. *Computers & Operations Research*, 35(1): 267–281, 2008.
2 citations sur les pages 155 et 157.
- [BW87] Judith O. BERKEY et Pearl Y. WANG : Two dimensional finite bin packing algorithms. *Journal of Operational Research Society*, 38: 423–429, 1987.
1 citation sur la page 14.
- [BZ08] Johannes BADER et Eckart ZITZLER : HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. TIK Report 286, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, novembre 2008.
2 citations sur les pages 41 et 143.
- [CCD⁺96] Jonathan CAGAN, Richard CLARK, Pratip DASTIDAR, Simon SZYKMAN et Paul WEISER : HVAC CAD layout tools: A case study of university/industry collaboration, 1996.
2 citations sur les pages 4 et 22.
- [CCWW00] Yun-Chih CHANG, Yao-Wen CHANG, Guang-Ming WU et Shu-Wei WU : B*-trees: a new representation for non-slicing floorplans. Dans *DAC ’00: Proceedings of the 37th conference on Design automation*, pages 458–463, New York, NY, USA, 2000. ACM.
1 citation sur la page 142.
- [CDY98] Jonathan CAGAN, Drew DEGENTESH et Su YIN : A simulated annealing-based algorithm using hierarchical models for general three-dimensional component layout. *Computer-Aided Design*, 30(10): 781–790, 1998.
1 citation sur la page 15.
- [CGJ83] Fan R. K. CHUNG, Michael R. GAREY et David S. JOHNSON : On packing two-dimensional bins. *SIAM Journal on Algebraic and Discrete Methods*, 3: 66–76, 1983.
1 citation sur la page 14.
- [CGO09] M. Teresa COSTA, A. Miguel GOMES et José F. OLIVEIRA : Heuristic approaches to large-scale periodic packing of irregular shapes on a rectangular sheet. *European Journal of Operational Research - Discrete Optimization*, 192: 29–49, 2009.
1 citation sur la page 15.

- [CJCM08] François CLAUTIAUX, Antoine JOUGLET, Jacques CARLIER et Aziz MOUKRIM : A new constraint programming approach for the orthogonal packing problem. *Computers and Operations Research*, 35(3): 944–959, 2008.
1 citation sur la page 141.
- [CJKO01] David W. CORNE, Nick R. JERRAM, Joshua D. KNOWLES et Martin J. OATES : PESA-II: Region-based selection in evolutionary multiobjective optimization. Dans *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, éditeurs : Lee SPECTOR, Erik D. GOODMAN, Annie WU, W. B. LANGDON, Hans-Michael VOIGT, Mitsuo GEN, Sandip SEN, Marco DORIGO, Shahram PEZESHK, Max H. GARZON et Edmund BURKE, pages 283–290, San Francisco, California, USA, juillet 2001. Morgan Kaufmann.
1 citation sur la page 143.
- [CKO00] David W. CORNE, Joshua D. KNOWLES et Martin J. OATES : The Pareto Envelope-based Selection Algorithm for multi-objective ptimisation. Dans *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 839–848, London, UK, 2000. Springer-Verlag.
1 citation sur la page 143.
- [CMWX08] Glauber CINTRA, Flávio K. MIYAZAWA, Yoshiko WAKABAYASHI et Eduardo C. XAVIER : Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research*, 191: 61–85, novembre 2008.
1 citation sur la page 12.
- [CS02] Yann COLLETTE et Patrick SIARRY : *Optimisation multiobjectif*. Numéro 978-2212111682. Eyrolles, 2002.
2 citations sur la page 26.
- [CSY02] Jonathan CAGAN, Kenji SHIMADA et Sun YIN : A survey of computational approaches to three-dimensional layout problems. *Computer-Aided Design*, 34(8): 597–611, 2002.
1 citation sur la page 21.
- [DA95] Kalyanmoy DEB et Ram Bhushan AGRAWAL : Simulated binary crossover for continuous search space. *Complex Systems*, 9: 115–148, 1995.
3 citations sur les pages 62 et 153.
- [DA99] Kalyanmoy DEB et Samir AGRAWAL : A niched-penalty approach for constraint handling in genetic algorithms. Dans *International Conference on Artificial Neural Networks and Genetic Algorithms*, éditeurs : R. ALBRECHT, A. DOBNIKAR, D. PEARSON et N. STEELE, pages 235–243. Springer-Verlag, 1999.
1 citation sur la page 153.
- [DAPM00a] Kalyanmoy DEB, Samir AGRAWAL, Amrit PRATAP et T. MEYARIVAN : A fast and elitist multi-objective genetic algorithm: NSGA-II. Rapport technique, Indian Institute of Technology, Kanpur Genetic Algorithms Laboratory (KANGAL), 2000.
2 citations sur les pages 34 et 37.
- [DAPM00b] Kalyanmoy DEB, Samir AGRAWAL, Amrit PRATAP et T. MEYARIVAN : A fast elitist Non-dominated Sorting Genetic Algorithm for multi-objective optimization: NSGA-II. Dans *Proceedings of the Parallel Problem Solving from Nature VI Conference*, éditeurs : Marc SCHOENAUER, Kalyanmoy DEB, Günter RUDOLPH, Xin YAO, Evelyne LUTTON, J. J. MERELO et Hans-Paul SCHWEFEL, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.
6 citations sur les pages 22, 34, 37, 39, 62 et 143.
- [Dar59] Charles DARWIN : *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, Albemarle Street, London, 1859.
1 citation sur la page 33.

-
- [Dav59] William C. DAVIDON : Variable metric method for minimization. Rapport technique ANL-5990, A.E.C. Research and Development, 1959.
1 citation sur la page 147.
- [DB99] Kalyanmoy DEB et Hans-Georg BEYER : Self-Adaptive: Genetic Algorithms with Simulated Binary Crossover. Rapport technique CI-61/99, Department of Computer Science, University of Dortmund, Germany, mars 1999.
1 citation sur la page 153.
- [DC03] Quan DING et Jonathan CAGAN : Automated trunk packing with extended pattern search. Dans *Proceedings of the 2003 SAE technical conferences*, Detroit, Michigan, USA, 2003.
1 citation sur la page 22.
- [DD92] Kathryn A. DOWSLAND et William B. DOWSLAND : Packing problems. *European Journal of Operational Research*, 56(1): 2–14, 1992.
1 citation sur la page 10.
- [Deb01] Kalyanmoy DEB : *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.
3 citations sur les pages 31, 33 et 35.
- [DF92] Harald DYCKHOFF et Ute FINKE : *Cutting and Packing in Production and Distribution: Typology and Bibliography*. Springer-Verlag, New York, 1992.
1 citation sur la page 10.
- [DFB06] Hong DONG, Georges M. FADEL et Vincent Y. BLOUIN : Vehicle component layout with shape morphing - an initial study. Dans *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Philadelphia, Pennsylvania, USA, septembre 2006.
1 citation sur la page 139.
- [DG01] Kalyanmoy DEB et Tushar GOEL : A hybrid Multi-Objective Evolutionary Approach to engineering shape design. Dans *First International Conference on Evolutionary Multi-Criterion Optimization*, éditeurs : Eckart ZITZLER, Kalyanmoy DEB, Lothar THIELE, Carlos A. Coello COELLO et David W. CORNE, pages 385–399. Springer-Verlag, 2001.
1 citation sur la page 38.
- [Dij59] Edsger W. DIJKSTRA : A note on two problems in connexion with graphs. *Numerische Mathematik*, 1: 269–271, 1959.
1 citation sur la page 49.
- [DJ96] Rahul DIGHE et Mark J. JAKIELA : Solving pattern nesting problems with genetic algorithms employing task decomposition and contact detection. *Evolutionary Computation*, 3(3): 239–266, 1996.
3 citations sur les pages 85, 86 et 92.
- [DJGM04] Kalyanmoy DEB, Prateek JAIN, N.K. GUPTA et H.K. MAJI : Multi-objective placement of electronic components using Evolutionary Algorithms. *IEEE Transactions on Components and Packaging Technologies*, 27(3): 480–492, 2004.
1 citation sur la page 20.
- [DMM03] Kalyanmoy DEB, Manikanth MOHAN et Shikhar MISHRA : A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. Rapport technique 2003002, KanGal, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, Kanpur, 2003, 2003.
1 citation sur la page 143.
- [DMM05] Kalyanmoy DEB, Manikanth MOHAN et Shikhar MISHRA : Evaluating the ϵ -domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. *Evolutionary Computation*, 13(4): 501–525, 2005.
1 citation sur la page 143.
-

- [DNLA09] Juan J. DURILLO, Antonio J. NEBRO, Francisco LUNA et Enrique ALBA : On the effect of the steady-state selection scheme in multi-objective genetic algorithms. Dans *EMO 09: Conference on Evolutionary Multi-Criterion Optimization* [EFG⁺09], pages 183–197. 1 citation sur la page 103.
- [DPHG07] Amine DRIRA, Henri PIERREVAL et Sonia HAJRI-GABOUJ : Facility layout problems: A survey. *Annual Reviews in Control*, 31(2): 255–267, 2007. 2 citations sur les pages 17 et 19.
- [DST97] Harald DYCKHOFF, Guntram SCHEITHAUER et Johannes TERNO : *Annotated Bibliographies in Combinatorial Optimization (M. Dell’Amico, F. Maffioli, Silvano Martello)*, chapitre Cutting and Packing (C&P), pages 393–413. Chichester, 1997. 1 citation sur la page 10.
- [DT08] Kalyanmoy DEB et Santosh TIWARI : Omni-Optimizer: A generic Evolutionary Algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185: 1062–1087, mars 2008. 8 citations sur les pages 36, 37, 62, 133, 143 et 154.
- [Dyc90] Harald DYCKHOFF : A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2): 145–159, janvier 1990. 8 citations sur les pages 4, 5, 9, 10, 11, 12 et 13.
- [EBPBA94] Ahmed EL-BOURI, Neil POPPLEWELL, Subramaniam BALAKRISHNAN et Attahiru S. ALFA : A search-based heuristic for the two-dimensional bin-packing problem. *INFOR special issue: Knapsack, packing and cutting, part II*, 32: 265–274., 1994. 1 citation sur la page 14.
- [EFG⁺09] Matthias EHRGOTT, Carlos M. FONSECA, Xavier GANDIBLEUX, Jin-Kao HAO et Marc SEVAUX, éditeurs. *Evolutionary Multi-Criterion Optimization, 5th International Conference, EMO 2009, Nantes, France, April 7-10, 2009, Proceedings*, volume 5467 de *Lecture Notes in Computer Science*. Springer, 2009. 2 citations sur les pages 176 et 184.
- [Ege03] Jens EGEBLAD : Placement Techniques for VLSI Layout Using Sequence-Pair Legalization. Mémoire de D.E.A., Department of Computer Science, University of Copenhagen, juillet 2003. 4 citations sur les pages 4, 20, 21 et 82.
- [EGL07] Hamidreza ESKANDARI, Christopher D. GEIGER et Gary B. LAMONT : FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems. Dans *EMO 07: Conference on Evolutionary Multi-Criterion Optimization*, éditeurs : Shigeru OBAYASHI, Kalyanmoy DEB, Carlo POLONI, Tomoyuki HIROYASU et Tadahiko MURATA, volume 4403 de *Lecture Notes in Computer Science*, pages 141–155. Springer, 2007. 1 citation sur la page 143.
- [Ehr00] Matthias EHRGOTT : *Multicriteria optimization*. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 2000. 1 citation sur la page 31.
- [EMH01] Mark ERICKSON, Alex MAYER et Jeffrey HORN : The niched Pareto genetic algorithm 2 applied to the design of the groundwater remediation systems. Dans *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO-2001)*, pages 681–695, 2001. 1 citation sur la page 34.
- [ENB09] Jens EGEBLAD, Benny K. NIELSEN et Marcus BRAZIL : Translational packing of arbitrary polytopes. *Comput. Geom. Theory Appl.*, 42(4): 269–288, 2009. 2 citations sur les pages 61 et 141.

-
- [ENO07] Jens EGEBLAD, Benny Kjær NIELSEN et Allan ODGAARD : Fast neighborhood search for two- and three-dimensional nesting problems. *European Journal of Operational Research*, 127(3): 1249–1266, décembre 2007.
3 citations sur les pages 15 et 141.
- [EP03] Jens EGEBLAD et David PISINGER : Heuristic approaches for the two-and three-dimensional knapsack packing problems. Rapport technique, DIKU, University of Copenhagen, Denmark, 2003.
1 citation sur la page 141.
- [EP09] Jens EGEBLAD et David PISINGER : Heuristic approaches for the two- and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36: 1026–1049, avril 2009.
1 citation sur la page 13.
- [FF93] Carlos M. FONSECA et Peter J. FLEMING : Genetic Algorithms for multi-objective optimization: Formulation, discussion and generalization. Dans *Fifth international conference on Genetic Algorithms*, pages 416–423, 1993.
4 citations sur les pages 30, 34 et 143.
- [FG87] Hans J. B. G. FRENK et Gábor GALAMBOS : Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem. *Computing*, 39(3): 201–217, 1987.
1 citation sur la page 14.
- [FH05] Efi FOGEL et Dan HALPERIN : Exact Minkowski sums of convex polyhedra. Dans *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 382–383, New York, NY, USA, 2005. ACM.
1 citation sur la page 86.
- [Fle70] Roger FLETCHER : A new approach to variable metric algorithms. *The Computer Journal*, 13(3): 317–322, août 1970.
2 citations sur les pages 62 et 147.
- [Fog62] Lawrence J. FOGEL : Autonomous automata. *Industrial Research*, 4(1): 14–19, 1962.
1 citation sur la page 143.
- [Fog88] David B. FOGEL : An evolutionary approach to the traveling salesman problem. *Biological Cybernetics*, 60(2): 139–144, 1988.
1 citation sur la page 143.
- [FP63] Roger FLETCHER et Mike J. D. POWELL : A rapidly convergent descent method for minimization. *Computer Journal*, 6: 163–168, 1963.
1 citation sur la page 147.
- [FPZ03] Oluf FAROE, David PISINGER et Martin ZACHARIASEN : Guided Local Search for final placement in VLSI design. *Journal of Heuristics*, 9(3): 269–295, 2003.
1 citation sur la page 141.
- [GCY99] Pei-Ning GUO, Chung-Kuan CHENG et Takeshi YOSHIMURA : An O-tree representation of non-slicing floorplan and its applications. Dans *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 268–273, New York, NY, USA, 1999. ACM.
2 citations sur les pages 21 et 142.
- [GDF73] Arthur M. GEOFFRION, James S. DYER et Andrew FEINBERG : An interactive approach for multicriteria optimization with an application to the operation of an accademic department. *Management Science*, 19: 357–369, 1973.
1 citation sur la page 110.
- [Gem74] Floyd W. GEMBICKI : *Vector optimization for control with performance and parameter sensitivity indices*. Thèse de doctorat, Case Western Reserve Univ., Cleveland, Ohio, 1974.
1 citation sur la page 31.
-

- [Geo68] Arthur M. GEOFFRION : Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Application*, 22: 618–630, 1968.
1 citation sur la page 31.
- [GF99] Pierre M. GRIGNON et Georges M. FADEL : Configuration Design Optimization Method. Dans *Proceedings of DETC'99 - ASME Design Engineering Technical Conferences*, page 11, Las Vegas, Nevada, 1999.
1 citation sur la page 135.
- [GF04] Pierre M. GRIGNON et Georges M. FADEL : A GA based configuration design optimization method. *Journal of Mechanical Design*, 126(1): 6–15, janvier 2004.
1 citation sur la page 22.
- [GG61] Paul C. GILMORE et Ralph E. GOMORY : A linear programming approach to the cutting stock problem. *Operations Research*, 9: 849–859, 1961.
1 citation sur la page 14.
- [GG63] Paul C. GILMORE et Ralph E. GOMORY : A linear programming approach to the cutting stock problem – part II. *Operations Research*, 11: 863–888, 1963.
2 citations sur les pages 12 et 14.
- [GJ79] Michael R. GAREY et David S. JOHNSON : *A guide to the theory of NP-completeness*. W.H. Freeman & Company, 1979.
2 citations sur la page 10.
- [GLM96] Stefan GOTTSCHALK, Ming C. LIN et Dinesh MANOCHA : OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30(Annual Conference Series): 171–180, 1996.
1 citation sur la page 142.
- [GMM90] Hermann GEHRING, Kerstin MENSCHNER et M. MEYER : A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, 44(2): 277–288, janvier 1990.
1 citation sur la page 13.
- [GO06] A. Miguel GOMES et José F. OLIVEIRA : Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, 171(3): 811–829, juin 2006.
5 citations sur les pages 102 et 141.
- [Goe92] Marc GOETSCHALCKX : An interactive layout heuristic based on hexagonal adjacency graphs. *European Journal of Operational Research*, 63: 304–325, 1992.
1 citation sur la page 44.
- [Gol70] Donald GOLDFARB : A family of variable metric updates derived by variational means. *Mathematics of Computing*, 24: 23–26, 1970.
2 citations sur les pages 62 et 147.
- [Gol89] David E. GOLDBERG : *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, janvier 1989.
3 citations sur les pages 30, 33 et 143.
- [GR87] David E. GOLDBERG et Jon RICHARDSON : Genetic algorithms with sharing for multimodal function optimization. Dans *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 41–49, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.
1 citation sur la page 35.
- [Gri98] Pierre M. GRIGNON : *Configuration Design Optimization Method*. Thèse de doctorat, Clemson University, 1998.
2 citations sur les pages 60 et 141.

- [GWF96] Pierre M. GRIGNON, John R. WODZIACK et Georges M. FADEL : Bi-Objective optimization of components packing using a genetic algorithm. Dans *NASA/AIAA/ISSMO Multidisciplinary Design and Optimization Conference*, pages 352–362, Seattle, Washington, septembre 1996. AIAA-96-4022-CP.
1 citation sur la page 7.
- [HCLC01] Sung-Min HUR, Kyung-Hyun CHOI, Seok-Hee LEE et Pok-Keun CHANG : Determination of fabricating orientation and packing in sls process. *Journal of Materials Processing Technology*, 112(2-3): 236 – 243, 2001.
1 citation sur la page 15.
- [HHC⁺00] Xianlong HONG, Gang HUANG, Yici CAI, Jiangchun GU, Sheqin DONG, Chung Kuan CHENG et Jun GU : Corner Block List: An effective and efficient topological representation of non-slicing floorplan. Dans *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 8–12, Piscataway, NJ, USA, 2000. IEEE Press.
1 citation sur la page 142.
- [HJ61] Robert HOOKE et Terry A. JEEVES : Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8(2): 212–241, 1961.
1 citation sur la page 22.
- [HK04] Nikolaus HANSEN et Stefan KERN : Evaluating the CMA Evolution Strategy on multimodal test functions. Dans *PPSN VIII: Conference on Parallel Problem Solving from Nature* [YBL⁺04], pages 282–291.
1 citation sur la page 143.
- [HMK03] Nikolaus HANSEN, Sibylle D. MÜLLER et Petros KOUMOUTSAKOS : Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1): 1–18, 2003.
2 citations sur les pages 36 et 143.
- [HNG94] Jeffrey HORN, Nicholas NAFPLIOTIS et David E. GOLDBERG : A niched pareto genetic algorithm for multi-objective optimization. Dans *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 82–87, 1994.
1 citation sur la page 143.
- [Hol75] John H. HOLLAND : *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.
2 citations sur la page 33.
- [HS87] Serge HUSTIN et Alberto SANGIOVANNI-VINCENTELLI : TIM, a new standard cell placement program based on the simulated annealing algorithm. Dans *IEEE Physical Design Workshop on Placement and Floorplanning*, Hilton Head, SC, avril 1987.
1 citation sur la page 21.
- [HSC04] Shinn-Ying HO, Li-Sun SHU et Jian-Hung CHEN : Intelligent Evolutionary Algorithms for large parameter optimization problems. Dans *IEEE Transactions on Evolutionary Computation*, volume 8, pages 522–541, 2004.
1 citation sur la page 143.
- [HT00] Edward HOPPER et Brian C. H. TURTON : An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1): 34–57, janvier 2000.
1 citation sur la page 142.
- [Hub96] Philip Martyn HUBBARD : Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3): 179–210, 1996.
2 citations sur les pages 94 et 142.

- [HYB07] Liao HUYAO, He YUANJUN et Julia A. BENNELL : The irregular nesting problem: a new approach for nofit polygon calculation. *Journal of the Operational Research Society*, 58(9): 1235–1245, septembre 2007.
1 citation sur la page 64.
- [IAG⁺08] Takashi IMAMICHI, Yohei ARAHORI, Jaeseong GIM, Seok-Hee HONG et Hiroshi NAGAMOCCHI : Removing overlaps in label layouts using multi-sphere scheme. Rapport technique 2008-006, Department of Applied Mathematics and Physics, Kyoto University and School of Information Technologies, University of Sydney, 2008.
1 citation sur la page 87.
- [IBK⁺97] Ilkka IKONEN, William E. BILES, Anup KUMAR, John C. WISSEL et Rammohan K. RAGADE : Genetic algorithm for packing three-dimensional non-convex objects having cavities and holes. Dans *Proceedings of the 7th International Conference on Genetic Algorithms*, éditeur : Thomas BÄCK, pages 591–598, East Lansing, Michigan, 1997. Morgan Kaufmann Publishers.
1 citation sur la page 15.
- [IN07] Takashi IMAMICHI et Hiroshi NAGAMOCCHI : A multi-sphere scheme for 2D and 3D packing problems. Dans *SLS*, pages 207–211, 2007.
1 citation sur la page 87.
- [IYN09] Takashi IMAMICHI, Mutsunori YAGIURA et Hiroshi NAGAMOCCHI : An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization*, 6(4): 345–361, novembre 2009.
4 citations sur les pages 15, 82, 85 et 141.
- [Jak96] Stefan JAKOBS : On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88(1): 165–181, janvier 1996.
3 citations sur les pages 7, 62 et 142.
- [Kan60] Leonid V. KANTOROVITCH : Mathematical methods of organizing and planning production. *Management Science*, 6: 366–422, 1960.
1 citation sur la page 14.
- [KC00] Joshua D. KNOWLES et David W. CORNE : Approximating the non-dominated front using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2): 149–172, 2000.
2 citations sur les pages 33 et 143.
- [KGP02] Nazan KHAN, David E. GOLDBERG et Martin PELIKAN : Multi-objective bayesian optimization algorithm. Rapport technique 2002009, Illinois Genetic Algorithms Laboratory (IlligAL), University of Illinois at Urbana-Champaign, Urbana, IL, USA, 2002.
1 citation sur la page 143.
- [KGV83] Scott KIRKPATRICK, Charles D. GELATT et Mario P. VECCHI : Optimization by simulated annealing. *Science*, 220(4598): 671–680, mai 1983.
4 citations sur les pages 15, 21, 102 et 142.
- [KHM⁺98] James T. KLOSOWSKI, Martin HELD, Joseph S. B. MITCHELL, Henry SOWIZRAL et Karel ZIKAN : Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4(1): 21–36, 1998.
1 citation sur la page 142.
- [KHMW04] Mifa KIM, Tomoyuki HIROYASU, Mitsunori MIKI et Shinya WATANABE : SPEA2+: Improving the performance of the Strength Pareto Evolutionary Algorithm 2. Dans *PPSN VIII: Conference on Parallel Problem Solving from Nature* [YBL⁺04], pages 742–751.
1 citation sur la page 143.

-
- [KL05] Saku KUKKONEN et Jouni LAMPINEN : GDE3: the third evolution step of generalized differential evolution. Dans *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 443–450, décembre 2005.
1 citation sur la page 143.
- [KPP04] Hans KELLERER, Ulrich PFERSCHY et David PISINGER : *Knapsack Problems*. Springer, Berlin, Germany, 2004.
1 citation sur la page 12.
- [Kuh55] Harold W. KUHN : The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2: 83–97, 1955.
1 citation sur la page 20.
- [Kur90] Frank KURSAWE : A variant of evolution strategies for vector optimization. *Parallel Problem Solving*, Nature I: 193–197, 1990.
2 citations sur les pages 34 et 143.
- [LC01] Jai-Ming LIN et Yao-Wen CHANG : TCG: A transitive closure graph-based representation for non-slicing floorplans. Dans *DAC '01: Proceedings of the 38th conference on Design automation*, pages 764–769, New York, NY, USA, 2001. ACM.
1 citation sur la page 142.
- [LHR02] Kyu-Yeul LEE, Seong-Nam HAN et Myung-Ii ROH : Optimal compartment layout design for a naval ship using an improved genetic algorithm. *Marine technology and SNAME news*, 39(3): 159–169, 2002.
1 citation sur la page 44.
- [LHR03] Kyu-Yeul LEE, Seong-Nam HAN et Myung-Il ROH : An improved genetic algorithm for facility layout problems having inner structure walls and passages. *Computers & Operations Research*, 30: 117–138, 2003.
2 citations sur les pages 44 et 50.
- [LHR05] Kyu-Yeul LEE, Seong-Nam HAN et Myung-Il ROH : An improved genetic algorithm for multi-floor facility problems having inner structure walls and passages. *Computers & Operations Research*, 32: 879–899, 2005.
1 citation sur la page 44.
- [LLM03] Lauro LINS, Sostenes LINS et Reinaldo MORÁBITO : An l -approach for packing (l, w) -rectangles into rectangular and l -shaped pieces. *Journal of the Operational Research Society*, 54: 777–789, 2003.
1 citation sur la page 15.
- [LM67] Robert C. LEE et James M. MOORE : CORELAP – COmputerized RELationship LAYOUT Planning. *Journal of Industrial Engineering*, 18(3): 195–200, 1967.
1 citation sur la page 44.
- [LM95] Zhenyu LI et Victor J. MILENKOVIC : Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operational Research*, 84(3): 539–561, août 1995.
1 citation sur la page 61.
- [LMV99] Andrea LODI, Silvano MARTELLO et Daniele VIGO : Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *SIAM Journal on Computing*, 11(4): 345–357, 1999.
1 citation sur la page 14.
- [LMV02] Andrea LODI, Silvano MARTELLO et Daniele VIGO : Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123(1–3): 379–396, 2002.
1 citation sur la page 14.
-

- [LN89] Dong C. LIU et Jorge NOCEDAL : On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3): 503–528, 1989.
1 citation sur la page 147.
- [LT99] Dequan LIU et Hongfei TENG : An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 112(2): 413–420, janvier 1999.
1 citation sur la page 142.
- [LT08] Zhan-wei LIU et Hong-fei TENG : Human-computer cooperative layout design method and its application. *Computers & Industrial Engineering*, 55: 735–757, novembre 2008.
1 citation sur la page 141.
- [MFG08] Yi MIAO, Georges M. FADEL et Vladimir B. GANTOVNIK : Vehicle configuration design with a packing genetic algorithm. *International Journal of Heavy Vehicle Systems*, 15: 433–448, décembre 2008.
4 citations sur les pages 4, 22 et 60.
- [MFNK96] Hiroshi MURATA, Kunihiro FUJIYOSHI, Shigetoshi NAKATAKE et Yoji KAJITANI : VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 15(12): 1518–1524, 1996.
2 citations sur les pages 5 et 142.
- [MI95] Tadahiko MURATA et Hisao ISHIBUCHI : MOGA: Multi-objective genetic algorithms. Dans *Second IEEE international conference on Evolutionary Computation*, pages 289–294, 1995.
2 citations sur les pages 34 et 143.
- [MP93] Giovanni MOSETTI et Carlo POLONI : Aerodynamic shape optimization by means of a genetic algorithm. Dans *5th International Symposium on Computational Fluid Dynamics*, Sendai, Japan, 1993.
1 citation sur la page 143.
- [Mun57] James MUNKRES : Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1): 32–38, mars 1957.
1 citation sur la page 20.
- [NFMK96] Shigetoshi NAKATAKE, Kunihiro FUJIYOSHI, Hiroshi MURATA et Yoji KAJITANI : Module placement on BSG-structure and IC layout applications. Dans *ICCAD '96: Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pages 484–491, Washington, DC, USA, 1996. IEEE Computer Society.
1 citation sur la page 142.
- [Nie07] Benny Kjær NIELSEN : An efficient solution method for relaxed variants of the nesting problem. Dans *CATS '07: Proceedings of the thirteenth Australasian symposium on Theory of computing*, éditeurs : Joachim GUDMUNDSSON et Barry JAY, volume 65 de *CRPIT*, pages 123–130. Australian Computer Society, Inc., 2007.
1 citation sur la page 15.
- [NM65] John A. NELDER et Roger R. MEAD : A simplex method for function minimization. *Computer Journal*, 7: 308–313, 1965.
1 citation sur la page 22.
- [NO03] Benny Kjær NIELSEN et Allan ODGAARD : Fast neighborhood search for the nesting problem. Rapport technique 03/03, Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark, 2003.
1 citation sur la page 15.
- [OI98] Takehiko ONO et Tsutomu IKEDA : Optimizing two-dimensional guillotine cut by genetic algorithms. *SIAM Journal of Optimization*, 8(3): 631–657, 1998.
2 citations sur les pages 21 et 142.

-
- [PG95] Ian J. PALMER et Richard L. GRIMSDALE : Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2): 105–116, juin 1995.
1 citation sur la page 142.
- [Pis02] David PISINGER : Heuristics for the container loading problem. *European Journal of Operational Research*, 141: 382–392, 2002.
1 citation sur la page 4.
- [Pis07] David PISINGER : Denser packings obtained in $O(n \log \log n)$ time. *INFORMS Journal on Computing*, 19(3): 395–405, 2007.
2 citations sur les pages 21 et 142.
- [PM09] Sergey POLYAKOVSKY et Rym M'HALLAH : An agent-based approach to the two-dimensional guillotine bin packing problem. *European Journal of Operational Research*, 192(3): 767–781, 2009.
1 citation sur la page 14.
- [PP98] Carlo POLONI et Valentino PEDIRODA : *Genetic algorithms and evolution strategy in engineering and computer science*, chapitre GA coupled with computationally expensive simulations: Tools to improve efficiency, pages 267–288. John Wiley & Sons, 1998.
1 citation sur la page 143.
- [PSG05] Martin PELIKAN, Kumara SASTRY et David E. GOLDBERG : Multiobjective hBOA, clustering, and scalability. Dans *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 663–670, New York, NY, USA, 2005. ACM.
1 citation sur la page 143.
- [PSL05] Kenneth V. PRICE, Rainer M. STORN et Jouni A. LAMPINEN : *Differential Evolution – A practical approach to global optimization*. Natural Computing Series. Springer-Verlag, Berlin, Germany, 2005.
1 citation sur la page 33.
- [Rec65] Ingo RECHENBERG : Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library Translation Number 1122, Farnborough, UK, 1965.
1 citation sur la page 33.
- [Rec73] Ingo RECHENBERG : *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Frommann-Holzboog Verlag, 1973.
1 citation sur la page 33.
- [Sch68] Hans-Paul SCHWEFEL : Projekt MHD-Straustrahlrohr: Experimentelle Optimierung einer Zweiphasendüse, Teil 1. Rapport technique 11.034/68, AEG Forschungsinstitut, 1968.
1 citation sur la page 33.
- [Sch84] James David SCHAFFER : *Some experiments in machine learning using Vector Evaluated Genetic Algorithms*. Thèse de doctorat, TN: Vanderbilt University, 1984.
2 citations sur les pages 34 et 143.
- [Sch92] Guntram SCHEITHAUER : Algorithms for the container loading problem. Dans *Operational Research*, éditeur : HEIDELBERG, pages 445–452. Springer-Verlag Berlin, 1992.
1 citation sur la page 13.
- [Sch95] Jason R. SCHOTT : Fault tolerant design using single and multicriteria genetic algorithm optimization. Mémoire de D.E.A., Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995.
1 citation sur la page 39.
- [SD94] N. SRINIVAS et Kalyanmoy DEB : Multi-objective optimization using Non-dominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3): 221–248, 1994.
1 citation sur la page 143.
-

- [SE67] Jerrold M. SEEHOF et Wayne O. EVANS : Automated layout design program. *Journal of Industrial Engineering*, 18: 690–695, 1967.
1 citation sur la page 44.
- [Sec88] Carl SECHEN : *VLSI placement and global routing using simulated annealing*. Kluwer Academic Publishers, 1988.
1 citation sur la page 21.
- [SGS⁺05] Yuriy STOYAN, Nikolay GIL, Guntram SCHEITHAUER, A. PANKRATOV et I. MAGDALINA : Packing of convex polytopes into a parallelepiped. *Optimization*, 54: 215–236, 2005.
1 citation sur la page 15.
- [Sha70] David F. SHANNO : Conditioning of quasi-newton methods for function minimization. *Mathematics of Computing*, 24: 647–656, 1970.
2 citations sur les pages 62 et 147.
- [SPGT98] Sanjay SACHDEV, Christiaan J. J. PAREDIS, Satyandra K. GUPTA et Sarosh N. TALUKDAR : 3D spatial layouts using A-teams. Dans *ASME Design Automation Conference*, pages 1–11, Atlanta, Georgia, USA, septembre 1998. ASME.
1 citation sur la page 141.
- [SPNE09] Ofer M. SHIR, Mike PREUSS, Boris NAUJOKS et Michael T. M. EMMERICH : Enhancing decision space diversity in evolutionary multiobjective algorithms. Dans *EMO 09: Conference on Evolutionary Multi-Criterion Optimization* [EFG⁺09], pages 95–109.
1 citation sur la page 36.
- [Sys91] Gilbert SYSWERDA : *Handbook of Genetic Algorithms 1*, chapitre Schedule Optimization using Genetic Algorithms, pages 332–349. Van Nostrand Reinhold, New York, 1991.
1 citation sur la page 50.
- [Tak00] Hideyuki TAKAGI : Active user intervention in an EC search. Dans *Proceedings of the 5th Joint Conf. on Information Sciences (JCIS2000)*, pages 995–998, 2000.
1 citation sur la page 139.
- [TFD09] Santosh TIWARI, Georges M. FADEL et Kalyanmoy DEB : AMGA2: Improving the performance of Archive-based Micro Genetic Algorithm for multi-objective optimization. Dans *IEEE Transactions on Evolutionary Computation*, 2009.
1 citation sur la page 143.
- [TFF08] Santosh TIWARI, Georges M. FADEL et Peter FENYES : A fast and efficient three dimensional compact packing algorithm for free-form objects. Dans *Proceedings of ASME International Design Engineering & Computers and Information in Engineering Conference (ASME DETC/CIE 2008)*, New York, NY, 2008.
8 citations sur les pages 4, 15, 17, 66, 81, 82, 142 et 169.
- [TFKD08] Santosh TIWARI, Georges M. FADEL, Patrick KOCH et Kalyanmoy DEB : AMGA: An Archive-based Micro Genetic Algorithm for multi-objective optimization. Dans *GEC-CO'08: Genetic and Evolutionary Computation Conference*, Atlanta, Georgia, juillet 2008.
2 citations sur les pages 38 et 143.
- [TFKD09] Santosh TIWARI, Georges M. FADEL, Patrick KOCH et Kalyanmoy DEB : Performance assessment of the hybrid Archive-based Micro Genetic Algorithm on the CEC09 test problems. Dans *IEEE Congress on Evolutionary Computation*. IEEE, 2009.
1 citation sur la page 143.
- [TN87] William C. THIBAUT et Bruce F. NAYLOR : Set operations on polyhedra using binary space partitioning trees. *SIGGRAPH Computer Graphics*, 21(4): 153–162, 1987.
1 citation sur la page 142.

-
- [TSqLL01] Hong-Fei TENG, Shou-Lin SUN, De quan LIU et Yan-Zhao LI : Layout optimization for the objects located within a rotating vessel – a three-dimensional packing problem with behavioral constraints. *Computers & Operations Research*, 28: 521–535, mai 2001.
3 citations sur les pages 4, 22 et 60.
- [TT97] Virginia TORCZON et Michael W. TROSSET : From evolutionary operation to parallel direct search: Pattern search algorithms for numerical optimization. *Computing Science & Statistics*, 29(1): 396–401, 1997.
2 citations sur les pages 22 et 142.
- [TTW00] Xiaoping TANG, Ruiqi TIAN et D. F. WONG : Fast evaluation of sequence pair in block placement by longest common subsequence computation. Dans *DATE '00: Proceedings of the conference on Design, Automation and Test in Europe*, pages 106–111, New York, NY, USA, 2000. ACM.
1 citation sur la page 142.
- [TV04] Mirela TĂNASE et Remco C. VELTKAMP : A straight skeleton approximating the medial axis. Dans *ESA*, éditeurs : Susanne ALBERS et Tomasz RADZIK, volume 3221 de *Lecture Notes in Computer Science*, pages 809–821. Springer, 2004.
1 citation sur la page 65.
- [Van99] David A. VAN VELDHUIZEN : *Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations*. Thèse de doctorat, Department of Electrical and Computer Engineering. Air Force Institute of Technology, Ohio, Wright-Patterson AFB, OH, 1999.
1 citation sur la page 41.
- [Wei07] Ron WEIN : Exact and approximate construction of offset polygons. *Computer-Aided Design*, 39(6): 518–527, 2007.
2 citations sur les pages 89 et 94.
- [WHK05] Ming-Jaan WANG, Michael H. HU et Meei-Yuh KU : A solution to the unequal area facilities layout problem by genetic algorithm. *Computers in Industry*, 56: 207–220, février 2005.
2 citations sur les pages 18 et 20.
- [WHM02] Shinya WATANABE, Tomoyuki HIROYASU et Mitsunori MIKI : Neighborhood Cultivation Genetic Algorithm for multi-objective optimization problems. Dans *Transactions of Information Processing Society of Japan*, volume 43, pages 183–198, 2002.
1 citation sur la page 143.
- [WHS07] Gerhard WÄSCHER, Heike HAUSSNER et Holger SCHUMANN : An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183: 1109–1130, décembre 2007.
8 citations sur les pages 5, 9, 10, 11, 15, 23 et 141.
- [YBL⁺04] Xin YAO, Edmund K. BURKE, José Antonio LOZANO, Jim SMITH, Juan J. Merelo GUERVÓS, John A. BULLINARIA, Jonathan E. ROWE, Peter TIÑO, Ata KABÁN et Hans-Paul SCHWEFEL, éditeurs. *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*, volume 3242 de *Lecture Notes in Computer Science*. Springer, 2004.
3 citations sur les pages 179, 180 et 186.
- [YC00] Su YIN et Jonathan CAGAN : An extended pattern search algorithm for three-dimensional component layout. *Journal of Mechanical Design*, 122(1): 102–108, mars 2000.
2 citations sur les pages 22 et 141.
- [YCH04] Su YIN, Jonathan CAGAN et Peter HODGES : Layout optimization of shapeable components with Extended Pattern Search applied to transmission design. *Journal of Mechanical Design*, 126: 188–190, 2004.
1 citation sur la page 22.
-

- [YH07] Taho YANG et Chih-Ching HUNG : Multiple-attribute decision making methods for plant layout design problem. *Robotics and Computer-Integrated Manufacturing*, 23: 126–137, 2007.
1 citation sur la page 57.
- [Yin00] Su YIN : *A computational framework for automated product layout synthesis based on extended pattern search algorithm*. Thèse de doctorat, Carnegie Mellon University, Pittsburgh, 2000.
3 citations sur les pages 4 et 22.
- [YK03] Taho YANG et Chunwei KUO : A hierarchical AHP/DEA methodology for the facilities layout design problem. *European Journal of Operational Research*, 147: 128–136, 2003.
1 citation sur la page 57.
- [YL03] Gary G. YEN et Haiming LU : Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. *IEEE Trans. Evolutionary Computation*, 7(3): 253–274, 2003.
1 citation sur la page 143.
- [ZDT00] Eckart ZITZLER, Kalyanmoy DEB et Lothar THIELE : Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2): 173–195, 2000.
2 citations sur les pages 40 et 41.
- [Zit99] Eckart ZITZLER : *Evolutionary Algorithms for multiobjective optimization: Methods and applications*. Thèse de doctorat, ETH Zurich, Switzerland, 1999.
1 citation sur la page 40.
- [ZK04] Eckart ZITZLER et Simon KÜNZLI : Indicator-based selection in multiobjective search. Dans *PPSN VIII: Conference on Parallel Problem Solving from Nature* [YBL⁺04], pages 832–842.
2 citations sur les pages 41 et 143.
- [ZLT01] Eckart ZITZLER, Marco LAUMANN et Lothar THIELE : SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. Dans *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, éditeurs : K. C. GIANNAKOGLU, D. T. TSAHALIS, J. PÉRIAUX, K. D. PAPAILIOU et T. FOGARTY, pages 95–100, Athens, Greece, 2001. International Center for Numerical Methods in Engineering.
2 citations sur les pages 34 et 143.
- [ZT98] Eckart ZITZLER et Lothar THIELE : An evolutionary algorithm for multiobjective optimization: The Strength Pareto approach. Rapport technique 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Gloriastrasse 35, CH-8092 Zurich, Switzerland, 1998.
1 citation sur la page 143.
- [ZTS08] Bao ZHANG, Hong-Fei TENG et Yan-Jun SHI : Layout optimization of satellite module using soft computing techniques. *Applied Soft Computing*, 8(1): 507–521, 2008.
3 citations sur les pages 4, 5 et 22.

Liste des tableaux

1.1	Classification des problèmes de découpe et de conditionnement proposée par Dyckhoff	12
1.2	Comparaison des problèmes C&P et des problèmes d'agencement	23
3.1	Représentation des données pour le problème d'agencement de navire	47
3.2	Paramètres de l'algorithme génétique utilisé	50
3.3	Comparaison des surfaces de compromis à l'aide de la métrique \mathcal{C}	54
4.1	Fonctions de minimisation testées pour le problème de séparation des boîtes	73
4.2	Résumé des caractéristiques des problèmes test pour le problème de séparation	73
4.3	Valeurs de la fonction de comparaison F_{comp} pour les différents problèmes de séparation	75
4.4	Distances entre solutions initiales et finales pour le problème de séparation	75
4.5	Résumé des différentes variantes de l'algorithme de séparation	100
5.1	Meilleurs résultats obtenus pour l'instance ALBANO	102
5.2	Coordonnées des polygones utilisés pour les problèmes d'agencements.	104
5.3	Propriétés des composants	104
5.4	Paramètres de l'algorithme génétique utilisé	106
5.5	Profilage du programme réalisé pour la résolution d'un problème d'agencement.	111
5.6	Récapitulatif des différentes simulations	116
5.7	Récapitulatif des valeurs obtenues pour l'hypervolume relatif	121
5.8	Comparaison des surfaces de compromis de \mathcal{S}_1 à \mathcal{S}_4 à l'aide de la métrique \mathcal{C}	121
5.9	Mesures de distance moyenne relatives	124
5.10	Mesures de représentation du front de Pareto	125
5.11	Mesures statistiques de la diversité phénotypique pour <i>NSGA-II</i> et <i>Omni-Optimizer</i>	125
A.1	Abréviations des problèmes de placement	141
A.2	Abréviations des méthodes de résolution des problèmes de placement.	141
A.3	Abréviations des heuristiques de placement et des schémas d'encodage	142
A.4	Abréviations des outils et représentations géométriques.	142
A.5	Abréviations des différents types de problèmes d'optimisation rencontrés.	142
A.6	Abréviations des différentes heuristiques et métaheuristiques.	142
A.7	Abréviations de différents algorithmes évolutionnaires.	143
C.1	Statistiques sur l'hypervolume relatif pour \mathcal{S}_1	163
C.2	Statistiques sur l'hypervolume relatif pour \mathcal{S}_2	164
C.3	Statistiques sur l'hypervolume relatif pour \mathcal{S}_3	164
C.4	Statistiques sur l'hypervolume relatif pour \mathcal{S}_4	164
C.5	Comparaison des surfaces de compromis de \mathcal{S}_1 à l'aide de la métrique \mathcal{C}	165
C.6	Comparaison des surfaces de compromis de \mathcal{S}_2 à l'aide de la métrique \mathcal{C}	165
C.7	Comparaison des surfaces de compromis de \mathcal{S}_3 à l'aide de la métrique \mathcal{C}	165
C.8	Comparaison des surfaces de compromis de \mathcal{S}_4 à l'aide de la métrique \mathcal{C}	166
C.9	Statistiques sur l'hypervolume relatif	166
C.10	Mesures statistiques sur les distances génotypiques moyennes	167
C.11	Mesures des distances phénotypiques maximales	167

C.12 Statistiques sur l'hypervolume relatif en fonction de p_m 168

C.13 Statistiques sur l'hypervolume relatif en fonction de p_s 168

Liste des figures

1.1	Présentation de quelques-uns des problèmes de découpe et de conditionnement	4
1.2	Présentation de quelques-uns des problèmes d'agencement	4
1.3	Fonction de pénalité pour un problème de chargement de palettes	6
1.4	Représentation de la fonction contrainte pour le problème de chargement de palettes .	7
1.5	Constituants principaux d'un problème de découpe et de conditionnement	9
1.6	Phénoménologie des problèmes de découpe et de conditionnement	11
1.7	Classification des problèmes de découpe et de conditionnement selon Wäscher <i>et al.</i> .	11
1.8	Solutions de deux problèmes de chargement de containers	13
1.9	Exemple de problème d'empaquetage (2DBP)	14
1.10	Mise en forme par procédé de frittage	16
1.11	Heuristique de placement <i>Bottom-Left</i> avec voxelisation des géométries	17
1.12	Exemple de résolution de problème de packing 3D	17
1.13	Constituants principaux d'un problème d'agencement	18
1.14	Structure des problèmes d'agencement d'ateliers	19
1.15	Utilisation des <i>Space Filling Curves</i>	20
1.16	Placements de composants pour l'intégration de composants à grande échelle	20
1.17	Agencement de 7 rectangles avec sa notation polonaise normalisée correspondante . .	21
2.1	Classification générale des méthodes d'optimisation	26
2.2	Espace des variables et d'espace d'arrivée	28
2.3	Illustration de quelques-unes des notions de dominance	29
2.4	Front de Pareto et front de Pareto faible	29
2.5	Illustration des différents fronts de Pareto pour un problème bi-objectif	29
2.6	Illustration des différentes définitions de rang	30
2.7	Illustration de la méthode des sommes pondérées pour un espace d'arrivée convexe . .	32
2.8	Illustration de la méthode des sommes pondérées pour un espace d'arrivée non convexe	32
2.9	Principe de fonctionnement des algorithmes génétiques	34
2.10	Déroulement de la procédure élitiste <i>NSGA-II</i>	35
2.11	Comparaison des mécanismes de diversité phénotypiques entre <i>NSGA-II</i> et <i>SPEA2</i> . .	36
2.12	Scénario où deux points efficaces adjacents sont éloignés dans l'espace de décision . . .	36
2.13	Distance phénotypique et génotypique	37
2.14	Distance phénotypique maximale	40
2.15	Calcul de l'hypervolume	41
2.16	Éléments nécessaires pour le calcul des métriques multi-objectifs	42
3.1	Vue 3D des différents ponts du navire considéré	46
3.2	Vue 2D des différents ponts du navire considéré	46
3.3	Densité des matrices de données d'adjacence et de flux	47
3.4	Encodage / Décodage des données pour l'agencement de compartiments d'un navire .	48
3.5	Différence entre calculs approché et réel des distances entre compartiment	49
3.6	Évolution des surfaces de compromis pour quatre optimisations	51
3.7	Solutions du problème d'agencement de navire	52
3.8	Évolution de l'hypervolume pour quatre optimisations	53

3.9	Évolution de la surface de compromis pour le problème d'agencement de navire	55
3.10	Évolution de la surface de compromis pour le problème d'agencement de navire	56
4.1	Algorithme génétique proposé pour la résolution des problèmes de placement	63
4.2	Composition de transformations pour le placement 2D	64
4.3	Calcul des polygones de non-recouvrement et d'appartenance	64
4.4	Axe médian et boules maximales	65
4.5	Heuristique <i>Left-Bottom-Fill</i> pour composants de géométries complexes	67
4.6	Profil des fonctions γ_{ij} et δ_{ij}	69
4.7	Profil des fonctions $\bar{\delta}_i$ et $\bar{\gamma}_i$	70
4.8	Exemples d'intersection de boîtes englobantes en 2D et 3D	72
4.9	Influence du choix de la fonction à minimiser pour le problème de séparation 2D – 40R	76
4.10	Influence du choix de la fonction à minimiser pour le problème de séparation 2D – 400R	77
4.11	Influence du choix de la fonction à minimiser pour le problème de séparation 2D – 1000R	77
4.12	Influence du choix de la fonction à minimiser pour le problème de séparation 3D – 30P	78
4.13	Influence du choix de la fonction à minimiser pour le problème de séparation 3D – 100P	79
4.14	Influence du choix de la fonction à minimiser pour le problème de séparation 2D – 300P	80
4.15	Présentation des six orientations possibles d'un parallélépipède dans l'espace	81
4.16	Présentation des quatre orientations pour une même boîte englobante	82
4.17	Calcul des profondeurs de pénétration en 2D	84
4.18	Calcul des contraintes d'appartenance g_i	84
4.19	Évolution de la fonction $f_{ij}(\mathbf{x})$ lors de l'entrée en collision des composants C_i et C_j .	85
4.20	Convergence de l'algorithme de séparation <i>ILSQN</i> sur l'instance <i>Dighe 2</i>	86
4.21	Minimisation de la somme des profondeurs de pénétration d'un assemblage de cercles .	90
4.22	Minimisation de la somme des profondeurs de pénétration d'un assemblage de cercles .	90
4.23	Minimisation de la somme des profondeurs de pénétration d'un assemblage de cercles .	91
4.24	Différents essais de décompositions de polygones en assembles de cercles.	91
4.25	Exemple d'échec pour l'algorithme de séparation avec une décomposition des composants	92
4.26	Convergence de l'algorithme <i>BFGS</i> pour le problème <i>Dighe 1</i>	92
4.27	Convergence de l'algorithme <i>BFGS</i> pour la séparation de 1000 sphères	95
4.28	Transformation d'un modèle 3D en assemblages hiérarchiques de sphères	95
4.29	Séparation en translation de 9 assemblages de sphères dans un parallélépipède	96
4.30	Détection de collisions à l'aide d'une représentation hiérarchique	97
4.31	Séparation en translation de 7 assemblages de sphères dans un contenant quelconque .	98
5.1	Comparaison des résultats sur l'instance Albano	103
5.2	Géométrie du contenant et des différents composants à positionner	104
5.3	Présentation des différents composants à positionner	105
5.4	Solutions extraites du front de deux surfaces de compromis	107
5.5	Évolution de la population dans l'espace des objectifs pour la 1 ^{ère} optimisation	108
5.6	Évolution de la population dans l'espace des objectifs pour la 2 ^{nde} optimisation	109
5.7	Premières solutions réalisables identifiées pour les 2 ^{ères} optimisations	110
5.8	Représentation des solutions efficaces à l'aide de la méthode <i>Path-value</i>	111
5.9	Front de Pareto obtenu pour le premier problème d'agencement	112
5.10	Solutions extraites du front de Pareto	113
5.11	Représentation des solutions efficaces du front de Pareto à l'aide de la méthode <i>Path-value</i>	114
5.12	Exemples de solutions sous-optimales	114
5.13	Influence de l'algorithme de séparation	115
5.14	Différence entre un algorithme pseudo-aléatoire et un algorithme évolutionnaire	116
5.15	Surfaces de compromis pour différentes probabilités de croisement	117
5.16	Surfaces de compromis de \mathcal{S}_1	118
5.17	Surfaces de compromis de \mathcal{S}_2	118
5.18	Surfaces de compromis de \mathcal{S}_3	119
5.19	Surfaces de compromis de \mathcal{S}_4	119

5.20	Comparaison des différentes surfaces de compromis avec le front de Pareto	120
5.21	Comparaison des valeurs de l'hypervolume relatif HVR	120
5.22	Comparaison statistique de l'évolution des valeurs de l'hypervolume relatif	122
5.23	Comparaison des hypervolumes relatifs pour les algorithmes <i>NSGA-II</i> et <i>Omni-Optimizer</i>	123
5.24	Comparaison de l'évolution des hypervolumes relatifs pour <i>NSGA-II</i> et <i>Omni-Optimizer</i>	124
5.25	Comparaison des distances génotypiques pour <i>NSGA-II</i> et <i>Omni-Optimizer</i>	126
5.26	Solutions efficaces pour une optimisation réalisée avec des solutions réalisables	127
5.27	Présentation des différents composants à positionner	129
5.28	Surfaces de compromis des 12 premières optimisations réalisées	129
5.29	Solutions extraites du front de Pareto pour le 2 nd problème d'agencement	130
5.30	Présentation de quelques-unes des solutions sous-optimales pour le problème de placement	131
5.31	Influence de la probabilité de mutation pour le problème d'agencement 2	132
5.32	Influence de la probabilité d'échange pour le problème d'agencement 2	132
5.33	Comparaison statistique de l'évolution des valeurs de l'hypervolume	133
5.34	Solutions extraites du front de Pareto	134
B.1	Illustration de la convergence de la méthode de Newton-Raphson	147
B.2	Densité de répartition de l'opérateur SBX	154
B.3	Densité de répartition de l'opérateur de mutation polynomiale	154
B.4	Calcul des polygones de non-recouvrement et d'appartenance	156
B.5	Calcul 2D de somme de Minkowski entre deux polygones	156
B.6	Construction du polygone de non-recouvrement pour deux polygones convexes	157
B.7	Décomposition d'un polygone en un ensemble de sous-polygones convexes	158
B.8	Calcul des domaines d'accessibilité 2D	160
B.9	Angle d'Euler $z - x - z$	161
D.1	Captures d'écran du démonstrateur développé	169
D.2	Architecture du démonstrateur développé	170

Liste des algorithmes

4.1	Évaluations de la fonction de séparation F et de son gradient ∇F	74
4.2	Évaluations de la fonction de séparation F et de son gradient ∇F pour les polygones .	85
B.1	Algorithme quasi-Newton <i>BFGS</i>	148
B.2	Pseudo-code de l'algorithme <i>Omni-Optimizer</i>	149
B.3	Sélection par tournoi pour un problème d'optimisation mono-objectif sous contraintes .	150
B.4	Sélection par tournoi pour un problème d'optimisation multi-objectif sous contraintes .	150
B.5	Procédure de sélection de deux individus proches dans l'espace des variables	151
B.6	Procédure de classement des solutions <i>ranking</i>	151
B.7	Procédure de calcul des distances entre solutions d'un même front	152

Table des matières

Sommaire	v
Nomenclature	vii
Introduction	1
1 Problèmes de placement : définitions et classifications	3
1.1 Introduction	3
1.2 Formulation des problèmes de placement	4
1.2.1 Définitions générales	4
1.2.2 Définition des problèmes de placement	5
1.2.3 Vocabulaire des problèmes de placement	5
1.3 Caractéristiques des problèmes de placement	5
1.3.1 Techniques de placement	5
1.3.2 Représentation des variables	6
1.3.3 Les objectifs des problèmes de placement	8
1.3.4 Les contraintes des problèmes de placement	8
1.3.5 Évaluation des contraintes de non-chevauchement et d'appartenance	8
1.4 Les problèmes de découpe et de conditionnement	9
1.4.1 Définition des problèmes de découpe et de conditionnement	9
1.4.2 Typologies des problèmes de découpe et de conditionnement	10
1.4.3 Problèmes de découpe	12
1.4.4 Problèmes de sac à dos	12
1.4.5 Le problème de <i>bin packing</i> à deux dimensions	13
1.4.6 Problème de chargement de palettes	15
1.4.7 Problème de découpe de formes irrégulières	15
1.4.8 Problèmes de conditionnement 3D d'objets de géométries quelconques	15
1.5 Les problèmes d'agencement	16
1.5.1 Problèmes d'agencement bidimensionnel	17
1.5.2 Problèmes d'agencement tridimensionnel	21
1.6 Comparaison des problèmes C&P et des problèmes d'agencement	22
1.7 Conseils pour la résolution d'un problème de placement	22
1.8 Conclusion	24
2 Modélisations et algorithmes d'optimisation pour les problèmes de placement	25
2.1 Introduction	26
2.2 Optimisation multi-objectif	27
2.2.1 Méthodes de résolution des problèmes multi-objectifs	31
2.3 Algorithmes évolutionnaires	31
2.3.1 Algorithmes génétiques	33
2.3.2 Algorithmes génétiques utilisés	37
2.3.3 Discussion	38
2.4 Évaluation de la qualité des surfaces de compromis	38

2.4.1	Indicateurs unaires	38
2.4.2	Indicateurs binaires	41
2.4.3	Discussion	42
2.5	Conclusion	42
3	Optimisation multi-objectif d'agencement de locaux	43
3.1	Introduction	44
3.2	Objectifs du problème	44
3.3	Étapes de la modélisation	45
3.3.1	Données du problème	45
3.3.2	Codage de l'information	46
3.3.3	Correspondance entre codage et agencement	47
3.3.4	Évaluation des individus	47
3.3.5	Calcul des distances entre compartiments	49
3.4	Optimisation multi-objectif	50
3.4.1	Variables d'optimisation	50
3.4.2	Choix de l'algorithme d'optimisation	50
3.4.3	Surfaces de compromis	50
3.4.4	Analyse des résultats	53
3.4.5	Extensions futures	54
3.5	Conclusion	57
4	Méthode de placement proposée	59
4.1	Introduction	60
4.2	Méthode de résolution des problèmes de placement proposée	60
4.2.1	Principe général	60
4.2.2	Modélisation du problème de placement bidimensionnel à orientations discrètes	63
4.2.3	Modélisation du problème de placement bidimensionnel à orientations continues	65
4.2.4	Modélisation du problème de placement tridimensionnel	65
4.2.5	Initialisation des solutions	66
4.3	Algorithmes de séparation	66
4.3.1	Algorithme de séparation en translation pour des composants parallélépipédiques	66
4.3.2	Algorithme de séparation en translation pour polygones	82
4.3.3	Algorithme de séparation pour des assemblages de cercles / sphères	87
4.3.4	Récapitulatif des différentes variantes de l'algorithme de séparation	100
4.4	Conclusion	100
5	Résolution de problèmes de placement bidimensionnels	101
5.1	Résultats obtenus sur un problème de découpe de formes irrégulières	102
5.2	Résultats obtenus pour un 1 ^{er} problème d'agencement	103
5.2.1	Données du problème	103
5.2.2	Premières optimisations	105
5.2.3	Solutions du front de Pareto	111
5.2.4	Rôle de l'algorithme de séparation	112
5.2.5	Rôle des opérateurs génétiques	115
5.2.6	Réglage des opérateurs génétiques	116
5.2.7	Choix de l'algorithme d'optimisation globale	123
5.2.8	Rôle de la population initiale	126
5.2.9	Discussion	128
5.3	Résultats obtenus pour un 2 ^{ème} problème d'agencement	128
5.3.1	Données du problème	128
5.3.2	Premières optimisations	129
5.3.3	Solutions du front de Pareto	131
5.3.4	Réglages des opérateurs génétiques	131

5.3.5	Relaxation des contraintes de placement	133
5.4	Conclusion	135
Conclusions et perspectives		137
A Abréviations		141
B Outils pour la résolution des problèmes de placement		145
B.1	Introduction	146
B.2	Optimisation continue sans contrainte	146
B.3	Optimisation multi-objectif	148
B.3.1	Procédures de l'algorithme <i>Omni-Optimizer</i>	148
B.3.2	Détails de certains opérateurs continus	153
B.4	Polygones de non-recouvrement et polygones d'appartenance	155
B.4.1	Polygones de non-recouvrement	156
B.4.2	Polygones d'appartenance	159
B.4.3	Remarques générales	159
B.5	Angle d'Euler	159
C Résultats annexes		163
C.1	Résultats annexes obtenus pour un 1 ^{er} problème d'agencement	163
C.1.1	Rôle des opérateurs génétiques	163
C.1.2	Choix de l'algorithme d'optimisation globale	166
C.2	Résultats annexes obtenus pour un 2 ^{ème} problème d'agencement	168
C.2.1	Réglages des opérateurs génétiques	168
D Présentation du démonstrateur développé		169
Bibliographie		171
Liste des tableaux		187
Liste des figures		189
Liste des algorithmes		193
Table des matières		195

Méthode générique pour l'optimisation d'agencement géométrique et fonctionnel

Guillaume JACQUENOT

Résumé

Dans de nombreux problèmes industriels, l'agencement des différents composants joue un rôle déterminant sur les performances du système à concevoir. Ces problèmes de placement ont fait l'objet de nombreux travaux dans la littérature ; toutefois les méthodes de résolution généralement proposées sont spécifiques et ne peuvent être appliquées à différents problèmes.

Ce travail propose une méthode générique pour la résolution des problèmes de placement. La méthode présentée est une hybridation d'un algorithme évolutionnaire avec une méthode de séparation. L'algorithme évolutionnaire est un algorithme génétique multi-objectif chargé d'explorer efficacement l'espace de recherche et l'algorithme de séparation a pour objectif de faire respecter les contraintes de placement du problème. Si les contraintes de placement d'une solution proposée ne sont pas respectées, l'algorithme de séparation modifie la solution de manière à la rendre réalisable. La méthode de séparation a été développée pour des cas simples comme pour des cas compliqués en 2D et 3D et permet la gestion de contraintes particulières.

Différents exemples 2D sont présentés avec plusieurs analyses permettant de comprendre les mécanismes mis en jeu lors de la résolution des problèmes de placement. Les éléments de résolution des problèmes 3D de géométries complexes sont aussi présentés. Par rapport aux algorithmes ad-hoc de la littérature, notre méthode générique permet de résoudre une grande variété de problèmes avec des temps de calculs du même ordre de grandeur. Enfin, les différents avantages et possibilités de la méthode permettent de nombreux développements futurs.

Mots-clés : Placement, Agencement, Méthode générique, Optimisation multi-objectif, Algorithme génétique, Algorithme de séparation.

Abstract

In several industrial problems, component layout plays a major role on the performance of the system to design. These placement problems have been the subject of several studies in literature, however the proposed methods developed are too specific and cannot be applied to other problems.

This thesis presents a generic method to solve multi-objective placement problem for free-form components. The proposed method is a relaxed placement technique combined with a hybrid algorithm based on an evolutionary algorithm and a separation algorithm. The evolutionary algorithm is a multi-objective genetic algorithm. The genetic algorithm is used as a global optimizer and is in charge of efficiently exploring the search space. The separation algorithm is used to legalize solutions proposed by the global optimizer, so that placement constraints are satisfied.

Different 2D test cases are presented with several analyses to understand the convergence mechanism. The different elements for solving the 3D regular and complex geometry problems are also presented. When compared to dedicated algorithms from literature, our generic method allows solving a wide range of problems with similar computation times. Finally, the different pros and possibilities of the method permit several future extensions.

Keywords : Placement problem, Layout problem, Cutting & Packing problem, Generic method, Multi-objective optimization, Genetic algorithm, Separation algorithm.

Discipline : Sciences de l'Ingénieur