Optimisation par essaim particulaire : adaptation de tribes à l'optimisation multiobjectif

Nadia Smairi

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
UNIVERSITE DE MANOUBA
ECOLE NATIONALE DES SCIENCES DE L'INFORMATIQUE

THÈSE

Présentée en vue de l'obtention du diplôme de

DOCTEUR EN INFORMATIQUE

Contribution à l'Optimisation par Essaim Particulaire : adaptation de TRIBES à l'optimisation multiobjectif

par
Nadia SMAIRI

Réalisée au sein de
Laboratoire de Stratégies d'Optimisation et Informatique intelligentE (SOIE), Université de Tunis
Laboratoire Images, Signaux et Systèmes Intelligents (LISSI), Université de Paris-Est

Soutenue le 06 Décembre 2013 devant le jury composé de :

Président :	Prof. Henda ben GHEZALA,	Université de Manouba
Rapporteur :	Prof. Saoussen KRICHEN,	Université de Jendouba
Rapporteur :	Prof. Jin-Kao HAO,	Université d'Angers
Directeurs de thèse :	Prof. Khaled GHEDIRA,	Université de Tunis
	Prof. Patrick SIARRY,	Université de Paris-Est

Remerciements

Ce travail a été effectué en cotutelle entre le laboratoire Soie de l'université de Tunis et le laboratoire LISSI de l'université de Paris-Est. Il m'est grandement agréable de rendre un sincère hommage à tous celles et ceux qui l'ont rendu possible par leur collaboration scientifique, par leurs soutiens et par leurs encouragements.

Je suis très honorée que Mme. Henda ben GHEZALA, Professeur à l'ENSI, ait accepté de présider le jury. Qu'elle soit assurée de mon profond respect.

Je remercie vivement Mme. Saoussen KRICHEN, Professeur à l'ISG de Tunis, et Mr. Jin-Kao HAO, Professeur à l'université d'Angers, d'avoir accepté de juger ce travail. Qu'ils soient assurés de ma profonde gratitude pour leurs précieux conseils qui m'ont permis d'améliorer le fond et la forme de mon manuscrit.

Je rends un grand hommage à Mr. Khaled GHEDIRA, Professeur à l'ISG de Tunis et à Mr. Patrick SIARRY, Professeur à l'université de Paris-Est, qui ont dirigé de près ce travail malgré leurs nombreuses occupations. Ils m'ont fait bénéficier de leurs compétences, de leurs conseils constructifs, de leurs encouragements et de leurs grandes qualités humaines. Je voudrai leur témoigner ma très profonde reconnaissance et ma très haute considération.

Sur le plan personnel, je tiens à exprimer mes remerciements et ma gratitude à tous les membres de la famille qui m'ont soutenue et encouragée sur tous les plans.

Que tous ceux que je n'ai pas cités, et ils sont certainement nombreux, se sentent associés à ces remerciements.

A la mémoire de ma très chère Maha.

Résumé

Dans le cadre de l'optimisation multiobjectif, les métaheuristiques sont reconnues pour être des méthodes performantes mais elles ne rencontrent qu'un succès modéré dans le monde de l'industrie. Dans un milieu où seule la performance compte, l'aspect stochastique des métaheuristiques semble encore être un obstacle difficile à franchir pour les décisionnaires. Il est donc important que les chercheurs de la communauté portent un effort tout particulier sur la facilité de prise en main des algorithmes. Plus les algorithmes seront faciles d'accès pour les utilisateurs novices, plus l'utilisation de ceux-ci pourra se répandre.

Parmi les améliorations possibles, la réduction du nombre de paramètres des algorithmes apparaît comme un enjeu majeur. En effet, les métaheuristiques sont fortement dépendantes de leur jeu de paramètres. Dans ce cadre se situe l'apport majeur de TRIBES, un algorithme mono-objectif d'Optimisation par Essaim Particulaire (OEP) qui fonctionne automatiquement, sans paramètres. Il a été mis au point par Maurice Clerc. En fait, le fonctionnement de l'OEP nécessite la manipulation de plusieurs paramètres. De ce fait, TRIBES évite l'effort de les régler (taille de l'essaim, vitesse maximale, facteur d'inertie, etc.).

Nous proposons dans cette thèse une adaptation de TRIBES à l'optimisation multiobjectif. L'objectif est d'obtenir un algorithme d'optimisation par essaim particulaire multiobjectif sans paramètres de contrôle. Nous reprenons les principaux mécanismes de TRIBES auxquels sont ajoutés de nouveaux mécanismes destinés à traiter des problèmes multiobjectif. Après les expérimentations, nous avons constaté, que TRIBES-Multiobjectif est moins compétitif par rapport aux algorithmes de référence dans la littérature. Ceci peut être expliqué par la stagnation prématurée de l'essaim. Pour remédier à ces problèmes, nous avons proposé l'hybridation entre TRIBES-Multiobjectif et un algorithme de recherche locale, à savoir le recuit simulé et la recherche tabou. L'idée était d'améliorer la capacité d'exploitation de TRIBES-Multiobjectif. Nos algorithmes ont été finalement appliqués sur des problèmes de dimensionnement des transistors dans les circuits analogiques.

Mots-clefs : optimisation, optimisation continue, optimisation multiobjectif, métaheuristiques, optimisation par essaim particulaire, algorithme adaptatif, TRIBES, recherche tabou, recuit simulé.

Abstract

Meta-heuristics are recognized to be successful to deal with multiobjective optimization problems but still with limited success in engineering fields. In an environment where only the performance counts, the stochastic aspect of meta-heuristics again seems to be a difficult obstacle to cross for the decision-makers. It is, thus, important that the researchers of the community concern a quite particular effort to ease the handling of those algorithms. The more the algorithms will be easily accessible for the novices, the more the use of these algorithms can spread.

Among the possible improvements, reducing the number of parameters is considered as the most challenging one. In fact, the performance of meta-heuristics is strongly dependent on their parameters values. TRIBES presents an attempt to remedy this problem. In fact, it is a particle swarm optimization (PSO) algorithm that works in an autonomous way. It was proposed by Maurice Clerc. Indeed, like every other meta-heuristic, PSO requires many parameters to be fitted every time a new problem is considered. The major contribution of TRIBES is to avoid the effort of fitting them.

We propose, in this thesis, an adaptation of TRIBES to the multiobjective optimization. Our aim is to conceive a competitive PSO algorithm free of parameters. We consider the main mechanisms of TRIBES to which are added new mechanisms intended to handle multiobjective problems. After the experimentations, we noticed that Multiobjective-TRIBES is not competitive compared to other multiobjective algorithms representative of the state of art. It can be explained by the premature stagnation of the swarm. To remedy these problems, we proposed the hybridization between Multiobjective-TRIBES and local search algorithms such as simulated annealing and tabu search. The idea behind the hybridization was to improve the capacity of exploitation of Multiobjective-TRIBES. Our algorithms were finally applied to sizing analogical circuits' problems.

Keywords: optimization, multiobjective optimization, continuous optimization, metaheuristics, particle swarm optimization, adaptive algorithm, TRIBES, tabu search, simulated annealing.

Sommaire

Introduction générale

Les chercheurs se heurtent souvent à des problèmes technologiques de complexité grandissante et qui peuvent être exprimés sous la forme d'un problème d'optimisation. Résoudre un problème d'optimisation consiste à trouver la ou les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur. Pour déterminer si une solution est meilleure qu'une autre, il est nécessaire que le problème introduise un critère de comparaison. Ainsi, la meilleure solution, appelée aussi solution optimale, est la solution ayant obtenu la meilleure évaluation au regard du critère défini. Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes appliqués : le traitement d'images, la conception de systèmes, la conception d'emplois du temps, ... La majorité de ces problèmes sont qualifiés de difficiles, car leur résolution nécessite l'utilisation d'algorithmes évolués, et en général il n'est pas possible de fournir dans tous les cas une solution optimale dans un temps raisonnable. Lorsqu'un seul critère est donné, par exemple un critère de minimisation de coût, la solution optimale est clairement définie, c'est celle qui a le coût minimal. Mais dans de nombreuses situations, un seul critère peut être insuffisant. En effet, la plupart des applications traitées intègrent plusieurs critères simultanés, souvent contradictoires. Intégrer des critères contradictoires pose un réel problème. La solution idéale n'existe pas, et il faut donc trouver un compromis. Dans ce cas, la solution optimale recherchée n'est plus un point unique, mais un ensemble de compromis. Résoudre un problème comprenant plusieurs critères, appelé communément problème multiobjectif, consiste donc à calculer le meilleur ensemble de solutions de compromis : le front Pareto. Dans cette thèse, nous traiterons principalement de la résolution de problèmes multiobjectif.

Dans ce cadre, l'étude des phénomènes réels est une source d'inspiration où l'étude et la modélisation des systèmes complexes sont très présentes. En recherche opérationnelle, cette approche a donné lieu à la naissance de nouvelles métaheuristiques. Les métaheuristiques forment une famille d'algorithmes stochastiques destinée à la résolution de problèmes d'optimisation difficile. Ces méthodes ont été conçues afin de résoudre un panel très large de problèmes, sans pour autant nécessiter de changements profonds dans les structures des algorithmes. Ces méthodes sont inspirées par des analogies avec la physique (recuit simulé), la génétique (algorithmes évolutionnaires) et l'éthologie (colonies de fourmis).

Parmi les méthodes inspirées de l'éthologie, la méthode d'Optimisation par Essaim Particulaire (OEP) a été proposée en 1995 par Kennedy et Eberhart pour la résolution de problèmes d'optimisation difficile continus. En effet, la plus grande particularité de l'OEP est le traitement des problèmes d'optimisation difficile à variables continues contrairement à la plupart des méthodes existantes qui ont été conçues pour traiter les problèmes discrets. L'idée motrice de cette technique est la coopération entre les différents éléments de l'essaim faisant ainsi émerger des comportements plus complexes.

Comme la majorité des métaheuristiques, l'OEP nécessite le réglage au préalable de plusieurs paramètres de contrôle en fonction du problème considéré. Les performances de l'OEP présentent une forte corrélation avec le réglage de ces paramètres. Il est donc indispensable pour un concepteur d'étudier l'influence de chaque paramètre sur le comportement de l'algorithme afin de déterminer le jeu de paramètres optimal. Cette étape est nécessaire chaque fois qu'un nouveau problème est considéré. Dans le cadre de l'industrie, l'OEP reste avec un succès limité. En effet, un ingénieur qui doit résoudre un problème d'optimisation n'a pas le temps de tester de nombreux jeux de paramètres avant de décider pour lequel opter. Ce constat a conduit à la nécessité du développement de méthodes adaptatives. Ces méthodes réduisent le nombre de paramètres d'un algorithme. En effet, elles incorporent des règles d'adaptation, qui permettent de modifier la valeur des paramètres, en fonction des résultats trouvés au cours du déroulement de l'algorithme. Le but final serait de concevoir des algorithmes sans paramètres de contrôle tout en garantissant leurs efficacités. Dans ce cadre, plusieurs applications ont été développées mais leurs utilisations restent restreintes à cause de la difficulté de leurs réglages surtout pour un utilisateur non spécialiste.

Cette thèse a été préparée au sein du laboratoire Soie de l'Université de Tunis en cotutelle avec le laboratoire Images, Signaux et Systèmes Intelligents de l'Université de Paris 12 Val de Marne.

Le but de cette thèse est l'étude de TRIBES, un algorithme d'Optimisation par Essaim particulaire sans paramètres, proposé par M. Clerc, initialement conçu pour l'optimisation mono-objectif. L'objectif premier est de présenter un algorithme d'optimisation multiobjectif adaptatif basé sur TRIBES et ceci à travers son adaptation à l'optimisation multiobjectif.

Ce mémoire, résumant notre contribution, comporte cinq chapitres organisés comme suit.

Dans le premier chapitre, nous présentons, tout d'abord, les concepts de base nécessaires à la bonne compréhension de notre travail. Nous détaillons en particulier l'optimisation multiobjectif.

Dans le deuxième chapitre, nous présentons un état de l'art des métaheuristiques d'optimisation multiobjectif. Une attention toute particulière est accordée à la méthode d'Optimisation par Essaim Particulaire, principale méthode utilisée au cours de cette thèse.

Le troisième chapitre est dévolu, en premier lieu, à une présentation de TRIBES. Nous passons par la suite à la présentation de notre contribution qui consiste à proposer une adaptation de TRIBES à l'optimisation multiobjectif, à savoir TRIBES-Multiobjectif. Cette étape sera couronnée par une comparaison de notre algorithme avec des algorithmes représentatifs de l'état de l'art. Le but est de soulever les avantages et les limites de notre proposition.

Dans le quatrième, nous décrivons en détail, notre deuxième contribution qui consiste à proposer des hybridations de TRIBES-Multiobjectif avec des techniques de recherche locale, à savoir la recherche tabou et le recuit simulé. En effet, nous avons proposé plusieurs schémas d'hybridation possibles.

Le cinquième et dernier chapitre sera consacré à l'application de nos algorithmes aux problèmes de dimensionnement des circuits. Nous considérons deux circuits, à savoir le convoyeur de courant de seconde génération et l'amplificateur faible bruit. Le but est de trouver les dimensions idéales des transistors formant ces circuits tout en optimisant leurs performances.

Enfin, nous présentons nos conclusions ainsi que des perspectives.

Chapitre 1 Concepts de base

1.1 Introduction

La résolution des problèmes d'optimisation est devenue un sujet central en recherche opérationnelle. En effet, le nombre de problèmes d'aide à la décision pouvant être formalisés sous la forme d'un problème d'optimisation est en forte croissance.

Dans ce chapitre, nous présentons un ensemble de définitions liées aux problèmes d'optimisation tout en mettant l'accent sur les problèmes d'optimisation multiobjectif. Nous exposons par la suite un large panel des méthodes de résolution existantes dans la littérature.

1.2 Problèmes d'optimisation mono-objectif

Résoudre un problème d'optimisation consiste à trouver une solution, appartenant à un espace de recherche *X*, qui minimise ou maximise un critère particulier *f*. Dans la plupart des cas, l'optimum découvert n'est pas unique. Ainsi il existe un ensemble de solutions minimisant ou maximisant le critère considéré. Dans la suite de ce document, nous aborderons les problèmes d'optimisation essentiellement sous l'aspect minimisation, maximiser une fonction *f* étant équivalent à minimiser *-f*. De plus, un problème d'optimisation peut présenter des contraintes d'égalité et/ou d'inégalité sur les solutions candidates $x \in X$.

Nous pouvons alors décrire formellement un problème d'optimisation comme suit :

$$\begin{cases} \text{Minimiser } f(x) \\ \text{Sous les contraintes } C_i(x) \quad \text{avec i de 1..m (m le nombre de contraintes)} \\ \text{Avec } x \in X \end{cases}$$

1.3 Vocabulaire et définitions

Dans cette section, nous détaillons les principaux concepts, hypothèses et notations sur lesquels nous nous sommes basés dans notre étude.

1.3.1 Vecteur des variables ou vecteur de décision

Il est formé par les différentes variables du problème, qui peuvent être de natures diverses. Ces variables expriment des données qualitatives ou quantitatives.

Considérons un problème ayant *n* variables. Dans ce cas le n-uplet $(x_1 x_2 .... x_n)$, formé par les variables du problème, définit son vecteur de variables x.

Ce vecteur peut prendre plusieurs valeurs définissant chacune une solution éventuelle.

1.3.2 Espace d'état

Il est défini par les domaines de définition des différentes variables du problème. Dans la plupart des problèmes, cet espace est fini.

1.3.3 Fonction objectif

Elle représente le but à atteindre par le décideur. C'est le nom donné à la fonction f (appelée aussi fonction de coût ou critère d'optimisation).

1.3.4 L'ensemble des contraintes

Il définit des conditions supplémentaires sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent sous forme d'égalité ou d'inégalité et permettent de limiter l'espace de recherche.

Les contraintes seront notées :

$C_i(x)$, avec i de 1 à m, avec m le nombre de contraintes.

1.4 Problèmes d'optimisation multiobjectif

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

Les problèmes d'optimisation multiobjectif sont une généralisation à *n* fonctions objectif des problèmes d'optimisation classiques. Ils sont définis formellement comme suit :

$$\begin{cases} \text{Minimiser } f(x) = (f_1(x), f_2(x), \dots, f_k(x)) \text{ (k le nombre d'objectifs)} \\ \text{Sous les contraintes } C_i(x) \quad \text{avec i de 1..m (m le nombre de contraintes)} \\ \text{Avec } x \in X \end{cases}$$

D'après cette définition, il est clair que l'optimum n'est plus une simple valeur comme pour les problèmes à un objectif, mais un ensemble de points, appelé l'ensemble des meilleurs compromis ou le front de Pareto.

Dans le cas multiobjectif, le concept d'optimum n'est pas le même que dans le cas mono-objectif. En effet, on n'est plus ici à la recherche d'un unique optimum global, mais plutôt d'une surface de solutions qui offrent un bon compromis entre les différents objectifs.

Définition 1: Relations d'ordre et de dominance :

Comme la solution optimale est une multitude de points, il est vital, pour identifier ces meilleurs compromis, de définir une relation d'ordre entre ces éléments. Dans le cas des problèmes d'optimisation multiobjectif, ces relations d'ordre sont appelées relations de dominance. Plusieurs relations de dominance ont déjà été présentées. Mais la plus célèbre et la plus utilisée est la dominance au sens de Pareto. C'est cette relation de dominance que nous allons définir et utiliser dans cette thèse. De manière à définir clairement et formellement cette notion, les relations =, ≤ et < usuelles sont étendues aux vecteurs.

Définition 2 (Principe de dominance): Soient u et v deux vecteurs de même dimension *n*

$u = v$ *ssi* $\forall i \in \{1, \dots, n\}, u_i = v_i$

$u \leq v$ *ssi* $\forall i \in \{1, \dots, n\}, u_i \leq v_i$

$u < v$ *ssi* $u \leq v$ et $u \neq v$

Les relations > et ≥ sont définies de manière analogue.

Les relations définies précédemment ne couvrent pas tous les cas possibles. En effet, il est impossible de classer les points a = (1, 2) et b = (2, 1) à l'aide d'une de ces relations.

Contrairement aux problèmes à un seul objectif où les relations usuelles <, ≤, . . . suffisent pour comparer les points, elles sont insuffisantes pour comparer des points issus de problèmes multiobjectif. Nous définissons donc maintenant la relation de dominance au sens de Pareto permettant de prendre en compte tous les cas de figures rencontrés lors de la comparaison de deux points (ici des vecteurs).

Définition 3 (ε-dominance) : Soient *u* et *v* deux vecteurs de même dimension *n*

u ε-domine *v* si et seulement si : $u_i \leq \varepsilon v_i \; \forall i \in \{1, ...,n\}$. Cette relation est notée $u \leq_\varepsilon v$.

Définition 4 (ε-dominance additive) : Soient *u* et *v* deux vecteurs de même dimension *n*
u ε-domine additivement *v* si et seulement si : $u_i \leq \varepsilon + v_i \ \forall i \in \{1, ...,n\}$. Cette relation est notée $u \leq_{\varepsilon+} v$.

Définition 5: Considérons un problème de minimisation. Soient u et v deux vecteurs de décision,

$u<v$ (*u* domine *v*)	*ssi* $f(u)<f(v)$
$u\leq v$ (*u* domine faiblement *v*)	*ssi* $f(u)\leq f(v)$
$u\sim v$ (*u* est incomparable (non dominé) avec *v*)	*ssi* $f(u)\nleq f(v)$et $f(v)\nleq f(u)$

Pour un problème de maximisation, ces relations sont définies de manière symétrique.
Un point *x* est dit Pareto-optimal s'il n'est dominé par aucun autre point de l'espace des solutions.
L'ensemble des points Pareto-optimaux définit la frontière de Pareto [Zitzler et al., 07].

Définition 6: Ensemble des solutions non dominées :

Soit F l'image dans l'espace des objectifs de l'ensemble des solutions réalisables X. L'ensemble des solutions non dominées de X, est défini par l'ensemble ND(X) :
ND(X)= $\{x \in X \lfloor x \ est \ non \ dominé \ par \ rapport$ à $X \}$
Le front de Pareto ND(F) de F est défini comme suit :
ND(F)= $\{y \in F \rfloor \nexists z \in F, z < y\}$

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

Un exemple de surface de compromis (front de Pareto) en dimension 2 est montré à la figure 1.1. Dans cet exemple, le problème considéré est un problème de minimisation avec deux critères.

1.5 Difficultés des problèmes multiobjectif

Les difficultés rencontrées lors de la résolution d'un problème multiobjectif sont, en plus de la présence de contraintes, liées aux propriétés des fonctions à optimiser. Ces dernières, selon la méthode d'optimisation employée, peuvent constituer des obstacles à la progression vers le front optimal global. Ces difficultés peuvent être regroupées en différentes catégories que nous proposons de présenter.

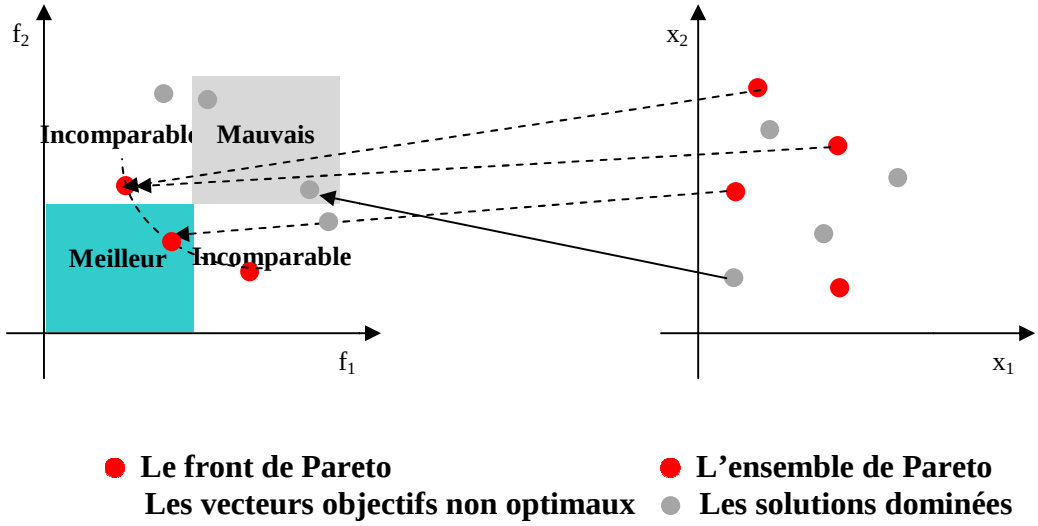


Figure 1.1 : Front de Pareto pour le cas bi-objectif

1.5.1 Convexité

Un front de Pareto P est dit convexe si, étant donné deux points distincts quelconques de ce front, le segment qui relie ces deux points est toujours contenu dans l'ensemble F des solutions atteignables de l'espace des objectifs (voir Figure 1.2).

La non convexité d'un espace entraîne l'existence de zones d'optimalité locale pouvant piéger les algorithmes de résolution (voir Figure 1.3).

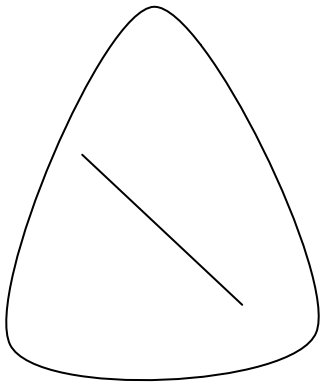


Figure 1.2 : Front convexe

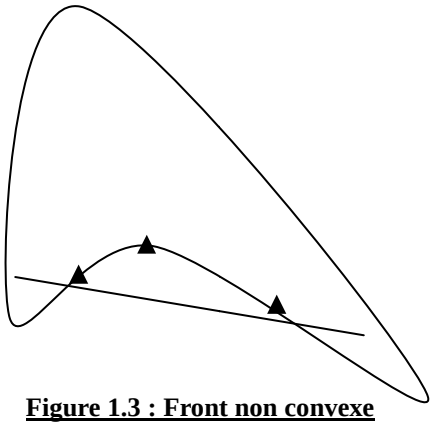


Figure 1.3 : Front non convexe

1.5.2 Multimodalité

La multimodalité du front de Pareto découle de l'existence de solutions localement Pareto- optimales. Les problèmes multimodaux sont caractérisés par l'existence de plusieurs fronts locaux successifs, qui sont susceptibles d'empêcher la détermination du front global. Ces fronts locaux agissent comme des attracteurs qui peuvent piéger les méthodes de résolution (voir Figure 1.4).

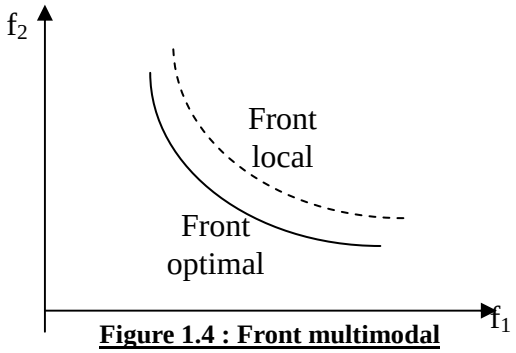


Figure 1.4 : Front multimodal

1.5.3 Discontinuité

La discontinuité d'un front peut également complexifier la résolution d'un problème multicritère. Cette discontinuité peut être liée à la présence de contraintes, ou à l'existence de zones optimales locales.

1.5.4 Non uniformité des solutions

Un problème d'optimisation multicritère peut être caractérisé par une distribution non uniforme des solutions dans l'espace des objectifs. Ainsi, certaines zones de l'espace des objectifs peuvent présenter des densités de solutions plus importantes que d'autres. Si les densités les plus fortes se situent loin du front optimal, les méthodes de résolution vont rencontrer plus de difficultés pour atteindre les solutions globales.

1.6 Les fonctions de test

La procédure de test utilisée pour comparer les algorithmes est celle qui a été définie dans le cadre de la conférence 2007 *IEEE Congress on Evolutionary Computation* (CEC'07). Le but de cette procédure est d'uniformiser les tests effectués sur les métaheuristiques et, ainsi, de faciliter les comparaisons entre celles-ci.

Tableau 1.1 : Les fonctions de Test

Fonction de test	Nombre des objectifs	Dimension de l'espace de décision	Multimodalité	Géométrie
Oka2	2	3	f_1 Unimodale f_2 Multimodale	Concave
Sympart	2	30	f_1 Multimodale f_2 Multimodale	Concave
S_ZDT1	2	30	f_1 Unimodale f_2 Unimodale	Convexe
S_ZDT2	2	30	f_1 Unimodale f_2 Unimodale	Concave
S_ZDT4	2	30	f_1 Unimodale f_2 Multimodale	Convexe
S_ZDT6	2	30	f_1 Multimodale f_2 Multimodale	Concave
R_ZDT4	2	10	f_1 Multimodale f_2 Multimodale	Concave
WFG1	3	24	f_1 Unimodale f_2 Unimodale f_3 Unimodale	Mixte
WFG8	3	24	f_1 Unimodale f_2 Unimodale f_3 Unimodale	Concave
WFG9	3	24	f_1 Multimodale f_2 Multimodale f_3 Multimodale	Concave

En effet, dans les différents articles sur le sujet, les algorithmes ne sont jamais évalués de la même façon, donc il est souvent difficile de comparer deux algorithmes sans avoir à les coder. Posséder une procédure de test commune permet donc de faciliter les comparaisons entre les différentes méthodes.

Lors de la conférence CEC'07 a été défini un ensemble de fonctions de test. Ces fonctions ont été choisies ou créées afin d'éviter certaines topologies de l'espace de recherche. En effet, il a été montré que certaines particularités topologiques des fonctions objectif peuvent favoriser certains algorithmes. Par exemple, il existe des algorithmes qui ont la capacité de converger plus rapidement si les solutions Pareto sont situées à l'origine [0, 0,…, 0] de l'espace de recherche ou au centre de celui-ci. Pour éviter cela, les fonctions définies lors de CEC'07 sont translatées par rapport à leur forme d'origine et/ou tournées. Certaines fonctions présentent aussi la particularité d'avoir les meilleures solutions situées sur une des frontières de l'espace de recherche.

Les fonctions du jeu de test CEC'07 représentent un large panel de difficultés à savoir, la convexité, la concavité, la multimodalité. Le tableau 1.1 résume les caractéristiques de chaque fonction.

1.7 Les métriques de comparaison

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

En effet, plusieurs aspects entrent en ligne de compte, notamment : la qualité (la proximité par rapport au front de Pareto théorique), la distribution (est-ce que toutes les parties du front de Pareto sont découvertes ?), la répartition (est-ce que les points sont répartis de manière homogène sur le front ?). Il est très difficile de prendre en compte tous ces paramètres au travers d'une seule valeur numérique. C'est pourquoi il est courant d'utiliser plusieurs métriques pour tester tel ou tel aspect de l'ensemble. De plus, il faut distinguer deux types de métriques : les métriques relatives, qui comparent deux ensembles, et les métriques absolues, qui évaluent un ensemble sans avoir besoin d'autres points ou ensemble de référence. Dans ce qui suit, nous présentons les principales métriques existant dans la littérature. Nous établissons par la même occasion un bilan des points faibles et des points forts pour chacune d'entre elles afin d'aider le lecteur à bien choisir les métriques qui correspondent au mieux à son problème.

1.7.1 La métrique d'espacement

Elle permet de mesurer l'uniformité de la répartition des points composant la surface de compromis. La définition de cette métrique est la suivante:

Soit P un ensemble de *n* éléments de dimension *m*, alors l'espacement de P que l'on note S(P) est:

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

Avec $d_i = min_j\left(\left|f_1^i - f_1^j\right| + ... + \left|f_m^i - f_m^j\right|\right)$ avec $i, j \in [1, n]$, $i \neq j$, $\overline{d}$ étant la moyenne de tous les d_i et f_k^i étant le coût de l'élément *i* suivant l'objectif *k*.

Elle permet de mesurer l'uniformité de la répartition des solutions dans l'espace des objectifs. Elle présente un moyen de mettre en avant l'aspect diversité d'un algorithme donné. Plus le résultat de la mesure est proche de 0, meilleure est la répartition des solutions. Dans le cas où le front est discontinu, l'interprétation de cette métrique devient difficile. En effet, les trous présents sur le front influent le calcul de S et les valeurs trouvées deviennent élevées.

1.7.2 Efficacité absolue

Elle calcule la proportion du front de Pareto P, trouvé par un algorithme donné, qui appartient au front de Pareto optimal P_{opt}. Pour ce faire, on doit connaître à priori le front de Pareto optimal du problème considéré. L'efficacité absolue que l'on note E_A est calculée de la manière suivante:

$$E_A = \frac{\left|P \cap P_{opt}\right|}{\left|P\right|}$$

Cette métrique met en avant la qualité des solutions trouvées. Plus le résultat de la mesure est proche de 1, meilleure est la qualité des solutions.

1.7.3 Le taux de succès

C'est une mesure du nombre de solutions trouvées qui appartiennent à l'ensemble de Pareto optimal. Il est calculé de cette manière: $TS = \sum_{i=1}^{n} s_i$ avec *n* le nombre de solutions trouvées, s_i est égale à 1 si la solution considérée est un membre de l'ensemble de Pareto optimal. Sinon, elle est égale à 0. Il est évident que le cas idéal est de trouver *TS* égal à *n*.

1.7.4 La covariance

C'est une métrique relative qui compare deux fronts *A* et *B*. La valeur *C*(*A*, *B*) correspond au pourcentage d'éléments de *B* qui sont dominés par, au moins, un des éléments de *A*. Le calcul de ce ratio s'effectue grâce à la formule suivante : $C(A,B) = \frac{\left|y \in B / \exists x \in A, y \text{ domine } x\right|}{\left|B\right|}$

Ainsi, $C(A, B) = 1$ implique que le front B est totalement dominé par A. A l'inverse, $C(A, B) = 0$ implique qu'aucun des points de B n'est dominé par un point de A. De ce fait, plus la valeur est proche de 1, meilleur sera considéré le front B par rapport à A. Cette métrique n'étant pas symétrique, il est nécessaire de considérer $C(A, B)$ et $C(B, A)$ pour obtenir une mesure plus fiable des deux fronts à comparer. Pour cela, il est préférable d'associer cette métrique à d'autres métriques comme celle de l'hypervolume pour différencier nettement deux fronts.

1.7.5 Largeur du front

La largeur du front, que l'on note *LF*, mesure la largeur du front de Pareto trouvé. *LF* est calculée suivant la formule suivante:

$$LF = \left(\sum_{u=1}^{k} \left(f_u(x_i) \right) \right)^{\frac{1}{2}}$$

1.7.6 Les indicateurs de qualité mesurant deux critères en même temps

Les critères présentés précédemment mesurent chacun un critère. Récemment la communauté de l'optimisation multiobjectif a vu apparaître de nouveaux indicateurs mesurant plusieurs critères simultanément. Nous présentons par la suite trois indicateurs mesurant la convergence et la diversité en même temps.

1.7.6.1 L'indicateur I_R

$$I_R = \frac{\sum_{\gamma \in B} u^*(\gamma, A) - u^*(\gamma, R)}{|B|}$$

avec:

R un front de référence.

A le front trouvé par l'algorithme proposé.

γ un vecteur de pondération.

u^* la valeur maximale atteinte par la fonction d'utilité u.

Dans le cas où l'on obtient des valeurs négatives, on conclut que le front trouvé est meilleur que le front de référence. Dans le cas où l'on obtient des valeurs très proches de zéro, on peut conclure que le front trouvé est très proche du front de référence [Knowles et al., 06].

1.7.6.2 L'indicateur d'Hypervolume I_H

L'indicateur d'Hypervolume mesure l'hypervolume de la portion de l'espace des objectifs qui est dominée par le front de Pareto trouvé: c'est un indicateur à maximiser. On peut utiliser une autre définition de l'indicateur d'Hypervolume, que l'on note I_H, calculant la différence d'hypervolume par rapport à un front de référence. Cet indicateur est alors à minimiser. Dans le cas où l'on obtient des valeurs négatives, on conclut que le front trouvé est meilleur que le front de référence. Dans le cas où l'on obtient des valeurs très proches de zéro, on peut conclure que le front trouvé est très proche du front de référence [Knowles et al., 06].

1.7.6.3 L'indicateur epsilon

L'indicateur Epsilon a été introduit dans [Knowles et al., 06] et comprend deux variantes : multiplicative et additive. Pour les deux versions il existe une forme unaire et une autre binaire. Cet indicateur est basé sur la notion de l'ε-dominance. La forme binaire multiplicative de l'indicateur epsilon, notée $I_\varepsilon(A,B)$, calcule la valeur minimum du facteur ε par laquelle chaque point dans B peut être multiplié, pour que l'ensemble résultant de cette transformation soit faiblement dominé par A :

$$I_\varepsilon(A,B) = inf_{\varepsilon\in\mathbb{R}}\{\forall z^2 \in B, \exists z^1 \in A: z^1 \preceq_\varepsilon z^2\}$$

Où z représente un vecteur objectif et $\preceq_\varepsilon$ la relation de l'epsilon-dominance multiplicative. De la même façon un indicateur unaire d'un ensemble A, $I_\varepsilon^1(A)$ peut être défini comme suit :

$$I_\varepsilon^1(A) = I_\varepsilon(A,R)$$

Où R représente un ensemble Référence. Par analogie, l'indicateur epsilon unaire additif $I_{\varepsilon+}^1(A)$ peut être obtenu en remplaçant la relation de l'ε-dominance multiplicative $\preceq_\varepsilon$ par celle de l'ε-dominance additive $\preceq_{\varepsilon+}$ dans l'équation précédente.

1.8 Les approches de résolution

La difficulté principale d'un problème d'optimisation multiobjectif est qu'il n'existe pas de définition de la solution optimale. Le décideur peut simplement exprimer qu'une solution est préférable à une autre, mais il n'existe pas de solution qui est meilleure que toutes les autres [Zitzler et al., 07]. Dans la littérature, il existe plusieurs méthodes de classification des

techniques d'optimisation multiobjectif. Dans ce qui suit, nous adoptons la classification qui se base sur la manière de traiter les objectifs : simultanément ou séparément.

1.8.1 Les méthodes agrégées

L'ensemble de ces méthodes reposent sur l'axiome suivant : tout décideur essaye d'optimiser une fonction d'utilité U qui est fonction de toutes les fonctions objectifs.

$$U= U(f_1, f_2, \ldots, f_k)$$

Les modèles courants les plus utilisés sont le modèle applicatif et le modèle additif.

L'utilisation de ces modèles impose que les objectifs soient exprimés dans une même unité. Il est très difficile d'utiliser ces techniques lorsque l'ensemble des objectifs sont de natures différentes.

1.8.2 La moyenne pondérée

Cette méthode consiste à additionner tous les objectifs en affectant à chacun d'eux un coefficient de poids. Ce coefficient représente l'importance relative attribuée à l'objectif. Cette méthode est simple à mettre en œuvre, mais ses difficultés concernent le choix de ces poids et l'expression de l'interaction entre les différents objectifs [Collette et al., 02].

1.8.3 Le but à atteindre

Le décideur fixe un but à atteindre T_i pour chaque objectif f_i. Ces valeurs sont ensuite ajoutées au problème comme des contraintes supplémentaires. La nouvelle fonction objectif est modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre. Cette méthode est aussi facile à mettre en œuvre, mais sa difficulté réside essentiellement dans la définition des objectifs à atteindre [Collette et al., 02].

1.8.4 La méthode ε-contrainte

Cette méthode est basée sur la minimisation d'un objectif f_i en considérant que les autres objectifs f_j avec $j \neq i$ doivent être inférieurs à une valeur ε_j. En général, l'objectif choisi est celui que l'on désire optimiser en priorité. Ce processus peut être itéré sur des objectifs différents, jusqu'à ce que l'on trouve des solutions satisfaisantes. La mise en œuvre de cette

méthode est simple, mais elle suppose la connaissance d'intervalles appropriés pour les valeurs de ε_j [Collette et al., 02].

1.8.5 Les méthodes non agrégées, non Pareto

Les méthodes basées sur une approche non Pareto ont pour caractéristique de traiter les objectifs séparément. Deux groupes de méthodes existent : les méthodes à sélection lexicographique et les méthodes à sélection parallèle.

Dans une approche classique de sélection lexicographique, les fonctions sont optimisées séquentiellement, suivant un ordre défini à priori. Cet ordre permet de définir le poids des objectifs. Plusieurs métaheuristiques ont été utilisées pour la résolution des problèmes multiobjectif à sélection lexicographique [Fourman, 85].

L'approche à sélection parallèle a été proposée par Schaffer. Son algorithme, inspiré d'un algorithme évolutionnaire et nommé VEGA, sélectionne les solutions courantes du front de Pareto suivant chaque objectif, indépendamment des autres (d'où le parallélisme). L'analyse du comportement de cet algorithme a montré qu'il se comportait d'une manière similaire à un algorithme utilisant une méthode agrégative [Schaffer, 84].

1.8.6 Approches Pareto

Contrairement aux deux approches précédentes, l'approche Pareto utilise la notion de dominance pour sélectionner des solutions faisant converger la population vers un ensemble de solutions qui approchent avec justesse le front de Pareto. Cette approche a été initiée par Goldberg en 1989 [Goldberg, 89]. Elle assure un traitement équitable de chaque critère, car il n'y a pas de classement à priori de l'importance des critères. Ainsi, en fin de traitement, l'algorithme fournit un ensemble de solutions qui approchent le front de Pareto. Le choix de la solution finale revient donc à l'utilisateur. Il doit choisir, parmi l'ensemble final, la solution qui lui convient le mieux. Ces méthodes se sont avérées être les plus efficaces. De nos jours, la majorité des algorithmes utilisent une approche Pareto pour traiter les problèmes multiobjectif. Cette famille d'approches peut être classifiée en deux grandes classes, à savoir les approches élitistes et les approches non élitistes.

1.8.6.1 Les techniques non élitistes

Ces techniques sont appelées non élitistes car elles ne conservent pas les individus Pareto-optimaux trouvés au cours du temps. De ce fait, il devient difficile de converger rapidement vers la frontière de Pareto et de garantir la diversité des solutions tout au long de cette frontière. Dans ce cadre, on peut citer plusieurs techniques qui existent dans la littérature, comme MOGA (*Multiple Objective Genetic Algorithm)* [Fonseca et al., 93].

1.8.6.2 Les techniques élitistes

Dans ces techniques, on introduit une population externe, ou archive, permettant de stocker les individus Pareto-optimaux. On utilise aussi des fonctions spécifiques (*clustering, crowding*) pour maintenir la diversité sur le front de Pareto [Berro, 01].

1.9 Conclusion

Nous avons présenté dans ce chapitre les principaux concepts de l'optimisation multiobjectif. Pour ce chapitre et pour le reste de cette thèse, nous nous plaçons dans le cadre de problèmes d'optimisation où il n'existe pas de modèle de préférences sur les critères (tous les critères sont de même importance). Toutes les problématiques liées à la modélisation des préférences, au choix de solutions au sein d'un ensemble de compromis, ou à la résolution interactive, sont traitées par la communauté d'aide à la décision dans un contexte multicritère.

Récemment les métaheuristiques, notamment les algorithmes évolutionnaires, ont permis la réalisation de méthodes de résolution dites Pareto très performantes. Dans le chapitre suivant, nous présentons les principales métaheuristiques appliquées dans le cadre de l'optimisation multiobjectif. Un intérêt particulier sera attribué à l'optimisation par essaim particulaire qui représente l'essentiel de notre sujet.

Chapitre 2 Métaheuristiques pour l’optimisation multiobjectif

2.1 Introduction

Tout comme pour l'optimisation mono-objectif, on peut distinguer deux grandes familles de méthodes de résolution pour traiter un problème multiobjectif : les méthodes exactes et les méthodes approchées. On remarque tout d'abord qu'il y a très peu de travaux sur les méthodes exactes dans le contexte de la résolution des problèmes d'optimisation multiobjectif difficiles. Sans doute, à cause de la grande difficulté de ce type de problème. La référence [Ehrgott et al., 00] présente la plupart des méthodes exactes existantes. Dans cette thèse, nous nous intéressons à la résolution approchée de problèmes d'optimisation multiobjectif notamment par des métaheuristiques. Ainsi, nous présentons brièvement dans ce chapitre deux types de métaheuristiques les plus connues : la recherche locale et les algorithmes évolutionnaires.

Pour une présentation plus élaborée des métaheuristiques, le lecteur est invité à consulter les références suivantes [Carlos et al., 00], [Coello, 99] et [Collette et al., 02]. Les métaheuristiques ont été appliquées avec succès sur un grand nombre de problèmes académiques et réels : problème d'affectation quadratique, coloriage de graphe, voyageur de commerce, etc.

2.2 Les méthodes de recherche locale

La première classe de métaheuristiques présentée regroupe les méthodes utilisant les principes de la recherche locale. Ces méthodes résolvent le problème d'optimisation de manière itérative. Elles font évoluer la configuration courante en la remplaçant par une autre issue de son voisinage. Ce changement de configuration est couramment appelé un mouvement. Nous commençons par présenter une étape commune à toutes les techniques de recherche locale, à savoir la détermination du voisinage, avant de passer à la présentation de quelques unes comme le recuit simulé et la recherche tabou.

2.2.1 Déterminer le voisinage

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

Le voisinage d'une solution courante x, que l'on note $V(x)$, est un sous-ensemble de solutions qu'il est possible d'atteindre par une série de transformations données. Il existe une

infinité de manières de choisir *V*, il faut adapter ce choix au problème, c'est-à-dire choisir la meilleure fonction *V* selon le problème considéré.

2.2.2 Le recuit simulé

Le recuit simulé s'inspire du processus du recuit thermique. Le processus du recuit simulé répète une procédure itérative qui cherche des configurations de coût plus faible, tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût [Kirkpatrick et al., 83].

L'algorithme type du recuit simulé pour un problème de minimisation est le suivant :

Algorithme 1 : Recuit Simulé

Paramètres d'entrées : T_0, α, Seuil, it_{palier}, *V*, f, x_0
$x \leftarrow x_0$
$T \leftarrow T_0$
Tant que T > Seuil **Faire**
  Pour nombre-itérations **de** 1 **à** it_{palier} **Faire**
    Choisir un point *x'* de *V*(*x*)
    $\Delta(f) \leftarrow f(x') - f(x)$
    Si $\Delta(f) < 0$ **Alors**
      $x \leftarrow x'$
    Sinon
      Choisir r de manière aléatoire entre 0 et 1
      Si $r < e^{(-\Delta(f) / T)}$ **Alors**
        $x \leftarrow x'$
      FinSi
    FinSi
  FinPour
  $T \leftarrow \alpha(T)$
FinTant que

Avec :

- T_0 la température initiale,
- *Seuil* le seuil minimal que la température peut atteindre,
- α la fonction diminuant la température lors du changement de palier,
- it_{palier} le nombre d'itérations à effectuer dans un palier,
- *V* la fonction de voisinage, *f* la fonction d'évaluation,
- x_0 la configuration initiale servant de point de départ à l'algorithme.

Dans cet algorithme, le nombre d'itérations (it_{palier}) devant être atteint pour effectuer un changement de palier est fixe. Dans certaines versions de cet algorithme, it_{palier} peut varier en fonction de la température (voir Algorithme 1).

2.2.3 La recherche Tabou

La recherche Tabou examine toutes les configurations appartenant à un voisinage $V(x)$ de x et retient la meilleure x_0 même si x_0 est plus mauvaise que x. Cependant, cette stratégie peut entraîner des cycles. Pour pallier ce problème, on mémorise les k dernières configurations visitées dans une mémoire à court terme et on interdit de retenir toute configuration qui en fait déjà partie. Cette mémoire, appelée la *liste Tabou*, est une des composantes essentielles de la méthode. En effet, elle permet d'éviter tous les cycles de longueurs inférieures à k, k étant un paramètre déterminé en fonction du problème [Glover et al., 97].

La mémorisation de configurations entières serait trop coûteuse en temps de calcul et en place mémoire. Il est préférable de mémoriser des caractéristiques des configurations au lieu de configurations entières. Plus précisément, il est possible de mémoriser uniquement un attribut de la configuration. Il en résulte que toutes les configurations possédant cet attribut, y compris celles qui n'ont pas encore été rencontrées, deviennent, elles aussi, tabou. Pour pallier à ce problème, un mécanisme particulier, appelé l'aspiration, est mis en place. Ce mécanisme consiste à révoquer le statut Tabou d'une configuration à certains moments de la recherche. La fonction d'aspiration la plus simple consiste à enlever le statut Tabou d'une configuration si celle-ci est meilleure que la meilleure configuration trouvée jusqu'alors (voir Algorithme 2).

Il existe aussi d'autres techniques permettant d'améliorer les performances de la méthode Tabou, en particulier, l'intensification et la diversification. L'intensification permet de se focaliser sur certaines zones de l'espace de recherche en apprenant des propriétés favorables, par exemple les propriétés communes souvent rencontrées dans les meilleures configurations visitées. La diversification est un processus inverse de l'intensification. En effet, elle cherche à diriger la recherche vers des zones inexplorées, en modifiant par exemple la fonction d'évaluation. L'intensification et la diversification jouent donc des rôles complémentaires [Glover et al., 97].

Une version classique de l'algorithme de la recherche Tabou est :

Algorithme 2 : Recherche Tabou

Paramètres d'entrées : V, f, x_0, L
$x \leftarrow x_0$
Pour i **de** 0 **à** L
  Choisir la meilleure configuration x' (x' un point de $V(x)$) autorisée
  Mettre à jour la liste Tabou en fonction de x'
  $x \leftarrow x'$

2.3 Les algorithmes évolutionnaires

Les algorithmes évolutionnaires, élaborés au cours des années 1950, forment une famille d'algorithmes de recherche inspirés de l'évolution biologique des espèces. L'idée ici est de s'inspirer de la théorie darwiniste de sélection naturelle pour résoudre des problèmes d'optimisation. On peut distinguer trois grandes classes d'algorithmes évolutionnaires : les algorithmes génétiques [Goldberg, 89], les stratégies d'évolution [Schwefel, 81] et la programmation évolutive [Fogel, 00].

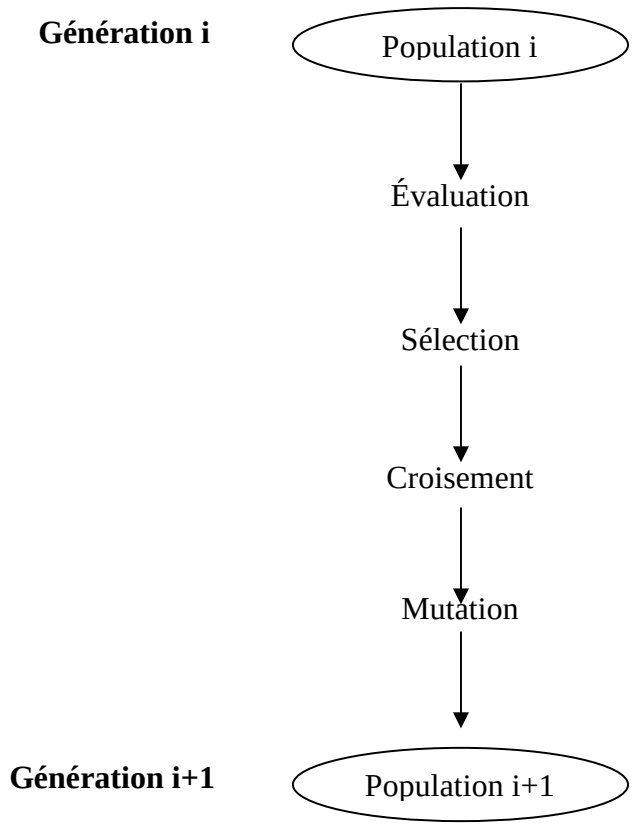


Figure 2.1 : Les étapes d'un algorithme évolutionnaire générique

Les approches évolutionnaires s'appuient toutes sur un modèle commun présenté par la figure 2.1. Les individus soumis à l'évolution sont des solutions possibles du problème posé. L'ensemble de ces individus constitue une population. Cette population évolue durant une succession d'itérations, appelées générations. Au cours de chaque génération, une succession d'opérateurs est appliquée aux individus de la population pour engendrer une nouvelle population, en vue de la génération suivante. Chaque opérateur utilise un ou plusieurs

individus de la population appelé(s) parent(s) pour engendrer de nouveaux individus, appelés enfants. A la fin de chaque génération, un certain nombre d'individus de la population sont remplacés par des enfants créés durant la génération. Un algorithme évolutionnaire dispose donc de trois opérateurs principaux : un opérateur de sélection, un opérateur de croisement et un opérateur de mutation. L'opérateur de sélection permet de favoriser la propagation des meilleures solutions dans la population, tout en maintenant la diversité génétique de celle-ci. L'opérateur de croisement est mis en œuvre lors de la phase de création des enfants. Le but de cet opérateur est d'échanger les gènes des différents parents pour créer les enfants.

L'opérateur de mutation consiste à tirer aléatoirement une composante de l'individu parent et à la remplacer par une valeur aléatoire. L'opérateur de mutation apporte un caractère aléatoire à la création de la descendance, qui permet de maintenir une certaine diversité dans la population.

Nous présentons par la suite quelques algorithmes évolutionnaires performants.

2.3.1 *Vector Evaluated Genetic Algorithm* (VEGA)

Cette méthode est une extension d'un algorithme génétique simple pour la résolution d'un problème mono-objectif. La seule différence avec un algorithme génétique simple est la manière dont s'effectue la sélection. En effet, si la population contient *n* individus, alors une sélection de *n/k* est effectuée pour chaque objectif. Ces *n/k* sous-populations sont ensuite mélangées afin d'obtenir une nouvelle population de taille *n.* le processus se termine par l'application des opérateurs génétiques de modification [Schaffer, 84]. Le schéma général de cet algorithme est donné par la figure 2.2.

Le problème de cette approche se situe essentiellement au niveau de la phase de sélection, qui ne tient compte que d'un seul objectif, elle ne privilégie que les individus qui obtiennent des bonnes performances pour cet objectif. Ceci va entraîner une élimination des individus moyens et une spécialisation des individus sélectionnés pour chaque objectif. Ce fait est contraire à l'objectif initial, qui essaye de trouver un compromis entre les différentes fonctions objectif.

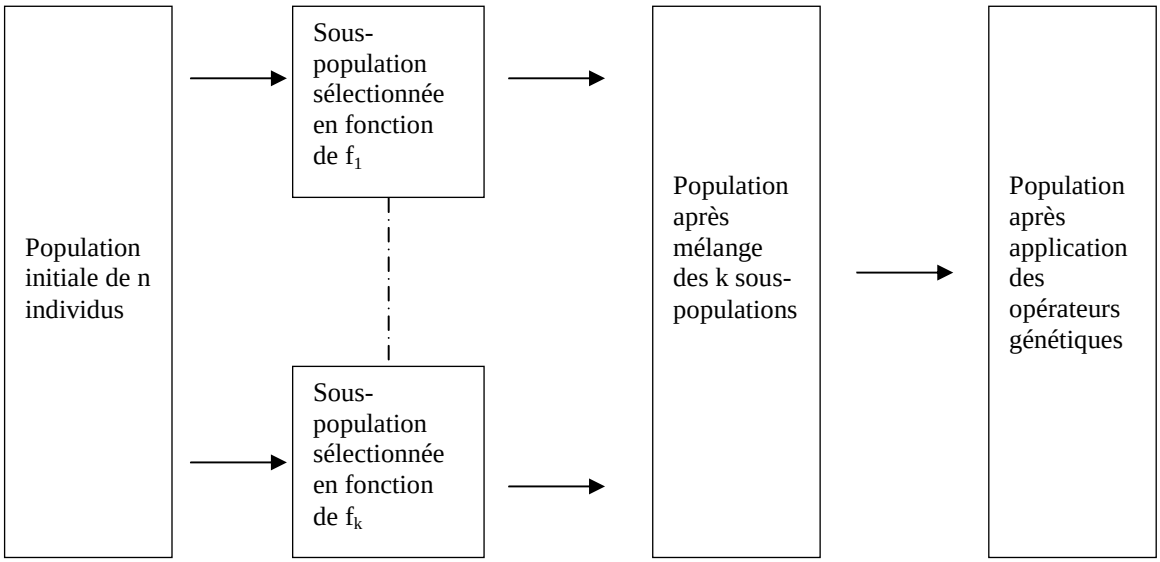


Figure 2.2 : Les étapes de l'algorithme VEGA

2.3.2 *Multiple Objective Genetic Algorithm* (MOGA)

Dans cette méthode, on introduit la notion de rang, qui permet de ranger les individus de la population suivant le nombre d'individus qu'ils dominent. La sélection est ensuite basée sur une notation qui prend en compte la fonction rang. Ce fait a tendance à répartir la population autour d'un même optimum. Pour éviter cet effet, on peut utiliser en plus une fonction de *sharing* pour répartir la population autour de la frontière de Pareto. En effet, cette fonction modifie la notation de l'individu en fonction de son isolement. En d'autres termes, un individu isolé avec une note moyenne sera préféré à un individu ayant une bonne note et se trouvant dans une zone peuplée [Fonseca et al., 93].

2.3.3 *Non Sorting Genetic Algorithm* (NSGA)

Cette technique est basée sur les algorithmes génétiques. Le passage d'une génération à une autre débute, tout d'abord, par une classification des individus. En effet, les individus non dominés forment le front F_1 et on leur applique une fonction de notation qui est supposée leur donner des chances égales pour se reproduire. Ensuite, l'ensemble F_1 est supprimé de la population. On recommence par la suite cette procédure afin de déterminer le front F_2 de la population, et la valeur de la fonction de notation pour ce groupe d'individus doit être inférieure à celle attribuée aux individus de F_1. Ce mécanisme est répété jusqu'à ce que l'on

ait traité tous les individus de la population. L'algorithme se déroule ensuite comme un algorithme génétique classique. Le grand inconvénient de cette méthode est sa grande complexité de calcul [Srivinas et al., 95].

2.3.4 Les techniques élitistes

Dans ces techniques, on introduit une population externe, ou archive, permettant de stocker les individus Pareto-optimaux. On utilise aussi des fonctions spécifiques (*clustering, crowding*) pour maintenir la diversité sur le front de Pareto [Berro, 01].

2.3.4.1 *Strength Pareto Evolutionary Algorithm* (SPEA)

Cette méthode représente une extension d'un algorithme génétique classique, qui consiste à comparer les solutions en se basant sur le concept de Pareto. Les solutions non dominées sont sauvegardées dans une archive externe. En effet, le passage d'une génération à une autre commence par la mise à jour de l'archive : tous les individus non dominés sont copiés dans l'archive et les individus dominés déjà présents sont supprimés. Si le nombre d'individus de l'archive excède un nombre donné, on applique une technique de *clustering* pour réduire sa taille. Ensuite, on applique la fonction de notation avant de passer à la sélection. Pour terminer, on applique les opérateurs génétiques de modification [Zitzler et al., 98].

2.3.4.2 *Pareto Archived Evolution Strategy* (PAES)

Cette technique n'est pas basée sur une population. Elle n'utilise qu'un seul individu à la fois pour la recherche des solutions. Celles qui sont temporairement Pareto-optimales sont stockées dans une archive externe. En effet, on commence par générer une solution candidate. Si elle est acceptée, alors on la sauvegarde dans l'archive, dans le cas où elle n'est pas dominée. Si l'archive devient pleine, alors on supprime un individu se situant dans la zone la plus peuplée. La mesure de l'encombrement se fait en se basant sur une fonction de *crowding* découpant l'espace des objectifs en des hypercubes [Knowles et al., 00].

2.3.4.3 *Non Sorting Genetic Algorithm 2* (NSGA-II)

C'est une amélioration de NSGA. En effet, pour comprendre le fonctionnement de l'algorithme, on se place à une génération *t*. Deux populations ($P_t$et $Q_t$de taille N) coexistent. La population P_t contient les meilleurs individus rencontrés jusqu'à la génération *t*, et la population Q_t est formée d'individus autres, issus des phases précédentes de l'algorithme. La

première étape consiste à créer la population R_t qui est l'union entre $P_t$et Q_t et à appliquer une procédure de *ranking* pour identifier les différents fronts F_i de solutions non dominées (les meilleurs individus se retrouvent donc dans le, ou les, tous premiers fronts). La deuxième phase consiste à construire une nouvelle population P_{t+1} contenant les N meilleurs individus de R_t. Il faut pour cela inclure intégralement les meilleurs fronts F_i (c'est-à-dire en commençant à l'indice 1) tant que le nombre d'individus présents dans P_{t+1} est inférieur à N. Il reste donc à ce stade N - | P_{t+1} | individus à inclure dans P_{t+1}.

Pour cela, une procédure de *crowding* est appliquée sur le premier front F_i non inclus. Les N - | P_{t+1} | meilleurs individus au sens de cette procédure de *crowding* sont insérés dans P_{t+1}. La troisième phase consiste à appliquer les opérations de sélection, de croisement et de mutation sur les individus de P_{t+1} pour obtenir la population Q_{t+1} [Deb et al., 02].

2.4 Les essaims particulaires

2.4.1 Présentation

Les essaims de particules forment une approche d'intelligence collective permettant de résoudre des problèmes d'optimisation. L'idée directrice de cette méthode est de simuler le comportement collectif des oiseaux à l'intérieur d'une nuée : leur capacité à voler de façon synchrone et leur aptitude à changer brusquement de direction, tout en restant en une formation optimale.

L'algorithme de cette approche, comme il a été développé, est basé sur un simple concept et peut être implémenté en quelques lignes de code. De plus, il n'utilise que des opérateurs mathématiques primaires (pas de gradients).

2.4.2 Essaim

Le terme essaim est basé sur cinq principes. Le premier principe est celui de proximité : la population devrait être capable d'effectuer de simples calculs de temps et d'espace. Le second est celui de qualité : la population devrait être capable de répondre aux facteurs de qualité dans l'environnement. Le troisième principe est celui de réponse diverse : la population ne devrait pas recommencer ses activités, excessivement, en suivant les mêmes chemins à chaque fois. Le quatrième est celui de stabilité : la population ne devrait pas changer son

mode de comportement chaque fois qu'il y a un changement d'environnement. Le dernier principe est celui d'adaptabilité : la population doit être capable de changer son mode de comportement quand cela en vaut la peine (en termes de complexité spatiale et temporelle).

2.4.3 Le voisinage

Le voisinage constitue la structure du réseau social. Les particules à l'intérieur d'un voisinage communiquent entre elles. Différents voisinages ont été étudiés et représentent essentiellement les topologies suivantes :

- Topologie en étoile : le réseau social est complet, chaque particule est attirée vers la meilleure particule notée p_g et communique avec les autres.
- Topologie en anneau : chaque particule communique avec *n* voisines immédiates. Chaque particule tend à se déplacer vers la meilleure dans son voisinage local notée p_l.
- Topologie en rayon : une particule centrale est connectée à toutes les autres. Seule cette particule ajuste sa position, si cela provoque une amélioration, l'information est communiquée aux autres.

2.4.4 La version standard de l'algorithme de l'optimisation par essaim de particules

Cette approche a été introduite par James Kennedy et Russel Eberhart en 1995. Elle part d'une population de particules ou d'individus se déplaçant dans l'espace de recherche. Le processus de recherche suit les règles suivantes [Clerc et al., 02] :

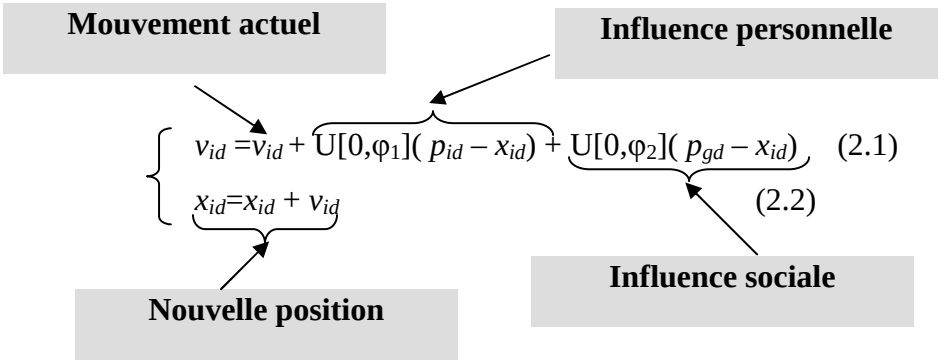
- Chaque particule est dotée d'une mémoire qui lui permet de mémoriser le meilleur point par lequel elle est déjà passée et elle a tendance à retourner vers ce point.
- Chaque particule appartient à un voisinage. Ce dernier a une influence sur son comportement. Cela signifie qu'une particule sera affectée par le meilleur point trouvé par n'importe quel membre de son voisinage topologique. Elle va tendre à aller vers ce point.

Pour cela, chaque particule est dotée d'une position, la valeur de la fonction objectif pour cette position, une vitesse, une mémoire personnelle retenant la meilleure position visitée et une mémoire collective retenant la meilleure position visitée par le voisinage. Les particules dans ce cas changent leur vitesse en se basant sur leur mouvement actuel, leur mémoire personnelle et leur mémoire collective. La position change en appliquant cette vitesse à la

position courante. De ce fait, le comportement de la particule est un compromis entre les trois possibilités suivantes :

- La particule suit son chemin personnel.
- La particule tend à retourner vers sa meilleure position.
- La particule tend à suivre la meilleure position trouvée par le voisinage.

Ce compromis est décrit par le formalisme suivant :



sachant que v_{id} est la vitesse de la particule i suivant la dimension d, x_{id} la position de la particule i suivant d, $U[0,\varphi_i]$ est une distribution entre une borne inférieure, qui est dans ce cas 0, et une borne supérieure, qui est dans ce cas φ_i. φ_1 et φ_2 représentent des accélérations pondérées mesurant l'influence de la mémoire personnelle et l'influence du voisinage de i.

Chaque particule représente une solution potentielle du problème d'optimisation. C'est le vecteur vitesse qui dirige le processus de recherche et reflète la sociabilité des particules.

Ce processus se répète tant que le critère de convergence n'est pas atteint. Cela peut être :

- Un nombre fixe d'itérations ;
- En fonction de la fonction objectif ;
- Lorsque la variation de vitesse est proche de 0.

Le déplacement d'une particule dans l'essaim peut être décrit par le schéma de la figure 2.3. Notons que ces équations de déplacement ne sont pas les seules. En effet, les stratégies de déplacement peuvent être divisées en deux catégories: une première catégorie contenant les stratégies utilisant une distribution indépendante pour chaque dimension (équations de déplacement classiques) et une deuxième catégorie contenant les stratégies qui dépendent uniquement des positions des particules.

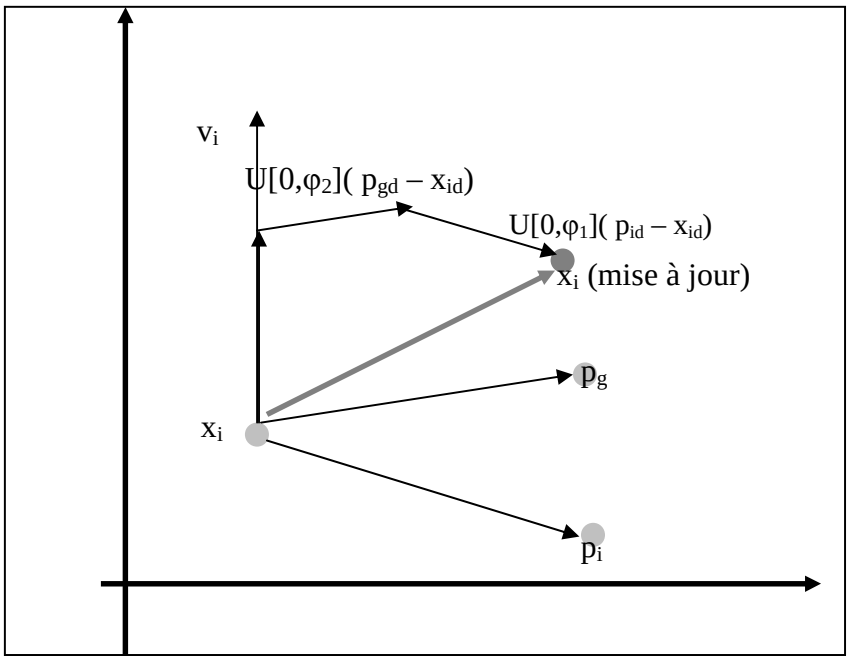


Figure 2.3 : Déplacement d'une particule

L'algorithme de l'OEP est caractérisé par l'interaction entre les différentes particules. Le mécanisme responsable de la génération de nouvelles solutions potentielles est l'imitation du comportement global du voisinage. L'algorithme 3 présente une version classique de l'algorithme de l'OEP [Kennedy, 97].

Le voisinage dans ce cas est un voisinage en étoile.

Nous constatons que les paramètres qui interviennent dans cet algorithme sont :

- La dimension du problème,
- Le nombre N de particules,
- Les valeurs des coefficients φ_1 et φ_2,
- La taille du voisinage,
- La vitesse.

2.4.5 La vitesse maximale

L'étude en profondeur de cet algorithme a démontré la tendance de quelques particules à subir une croissance explosive au niveau de la vitesse. Pour remédier à ce problème, un seuil arbitraire V_{max} est placé sur l'ampleur de la vitesse de n'importe quelle particule. Cependant, la valeur de V_{max} doit être suffisamment grande pour ne pas tomber dans le piège des minima locaux.

Algorithme 3 : OEP classique

N le nombre de particules
F la fonction objectif
[**Initialisations**]
//Initialiser aléatoirement la population
Init_population(P)
// Evaluer la performance de chaque particule dans la population
Evaluer(P)
[**Traitements**]
Répéter
  Pour i de 1 à N
    // trouver le voisinage de i
    V← **obtenir-voisinage(i)**
    // sélectionner les sources d'influence du voisinage
    Source ← **selection-source(i,V)**
    // Générer une nouvelle position en se basant sur les sources d'influence
    x ←**Generer-nouvelle-pos(Source)**
    Evaluer(x)
   // Tester si la nouvelle position est meilleure que l'ancienne meilleure position déjà visitée
      Si F(pi) est meilleur que F(x)
        pi ←x
      Fin si
  Fin Pour
Jusqu'à ce que le processus converge

De ce fait (2.1) devient :

Si $v_{id} > V_{max_d}$ **alors** $v_{id} = V_{max_d}$ et **Si** $v_{id} < -V_{max_d}$ **alors** $v_{id} = -V_{max_d}$.

En effet, les différentes expériences réalisées ont démontré que les paramètres φ_1 et φ_2 peuvent être fixés à 2 pour toutes les applications. V_{max} est le seul paramètre qui a eu besoin d'être ajusté. Il est cependant démontré que ce choix dépend des problèmes traités. Il n'y a pas un choix optimal de valeur qui doit être affecté à V_{max} [Angeline, 98].

2.4.6 Le contrôle de position

La plupart des recherches conduites sur l'OEP cherchent la valeur optimale de la fonction objectif dans un certain hypercube. Il est donc nécessaire, d'une façon ou d'une autre, de favoriser l'exploration pour rester à l'intérieur de cet hyperespace valide. Ceci est souvent réalisé en remettant les particules dans des limites valables, chaque fois que c'est nécessaire. Cependant, dans quelques situations, la réinitialisation fait plus de mal que de bien.

2.4.7 L'inertie

Les chercheurs sont insatisfaits de l'effet et du choix arbitraire de V_{max}. Il est spécifique au problème et aucune règle générale ne semble exister. De plus, avec son implémentation, l'algorithme ne converge pas. Il favorise uniquement l'exploitation plus que l'exploration. Pour réduire et éliminer l'influence de la vitesse maximale, on a également introduit un facteur d'inertie *w*. Dans ce cas, (2.1) devient :

$$v_{id} = \boldsymbol{w} v_{id} + U[0,\varphi_1]\,(p_{id} - x_{id}) + U[0,\varphi_2]\,(p_{gd} - x_{id})$$

En effet, un grand facteur d'inertie induit une grande exploration de l'espace de recherche alors qu'un petit facteur concentre la recherche sur un petit espace. On a étudié deux implémentations possibles de ce facteur :

- Constante,
- Une fonction linéaire décroissante du temps.

Le but de l'utilisation de la fonction décroissante en fonction du temps est de permettre à l'essaim de couvrir un grand secteur de l'espace de recherche par une recherche grossière au début. Par la suite, le facteur ***w*** commence à amortir progressivement la vitesse des particules, les forçant ainsi à effectuer une recherche locale plus fine.

L'expérience a montré qu'en utilisant le facteur d'inertie, V_{max} pourrait simplement être mis à la valeur de la gamme dynamique de chaque particule (suivant chaque dimension).

2.4.8 Coefficient de resserrement

Un autre moyen pour contrôler la vitesse est l'ajout d'un coefficient de restriction *K*, qui élimine la tendance de certaines particules à se déplacer rapidement d'une région à une autre. Dans ce cas, (2.1) devient :

$$v_{id} = \boldsymbol{K}(v_{id} + U[0,\varphi_1]\,(p_{id} - x_{id}) + U[0,\varphi_2]\,(p_{gd} - x_{id}))$$

avec $K = \dfrac{2}{\left|2 - \rho - \sqrt{\rho^2 - 4\rho}\right|}$ et $\rho = \rho_1 + \rho_2$, $\rho > 4$.

Dans ce cas, V_{max} n'est pas obligatoire.

2.4.9 Les *NoHopes* essais et les processus d'amélioration

Les *NoHopes* essais sont des tests à effectuer pour savoir si l'amélioration de la solution est impossible dans les itérations suivantes.

2.4.9.1 Les *NoHopes* essais [Clerc, 99]

- **Critère1**

« L'essaim est trop petit ». On est obligé de calculer le diamètre de l'essaim à chaque itération. En effet, quand « la distance » entre deux particules devient trop petite, elles deviennent identiques (tout d'abord au niveau de la position, ensuite au niveau de la vitesse).

- **Critère 2**

« L'essaim est trop lent ». On compare les vitesses des particules à un seuil (individuellement ou globalement). Ce seuil peut être modifié à chaque itération de l'algorithme.

- **Critère 3**

« Pas d'améliorations pendant les dernières itérations ».

Expérimentalement, il s'avère que les critères 1 et 2 sont suffisants.

2.4.9.2 Les processus d'amélioration

Une fois que l'un de ces critères est vérifié, il faut appliquer une des stratégies suivantes pour améliorer les solutions trouvées.

- **Méthode de descente directe**

Chaque particule retourne à sa dernière meilleure position. Elle effectue des mouvements lents et arbitraires et s'arrête si elle trouve une meilleure position, ou si elle dépasse un nombre de mouvements N_{max}, fixé au préalable. Si l'essaim obtenu dans ce cas est plus petit que l'initial, il est alors complété par des particules arbitrairement choisies.

- **Méthode de descente profonde**

Chaque particule retourne à sa dernière meilleure position. Puis, elle effectue des mouvements lents tant qu'elle trouve des meilleures positions (le nombre de mouvements ne doit pas dépasser N_{max}, fixé au préalable). Si l'essaim obtenu dans ce cas est plus petit que l'initial, il est alors complété par des particules arbitrairement choisies.

2.4.10 L'application de l'OEP à un problème

Afin d'utiliser l'OEP efficacement, plusieurs paramètres sont mis en jeu. Pour cela, ils doivent être définis avec succès. Parmi ces paramètres, on peut citer essentiellement :

- **Le codage des solutions :** Le type de codage dépend du problème. Les solutions sont généralement codées comme étant des vecteurs appartenant à un espace vectoriel de dimension D, qui est la dimension du problème. Cela induit la définition de deux bornes (inférieure et supérieure) suivant chaque dimension. On définit ainsi un hypercube.
- **La fonction objectif :** Elle dépend également du problème. Elle représente un moyen pour mesurer la qualité de chaque solution. Cette fonction doit être capable de créer un ordre total dans l'espace des solutions. Il est aussi important de signaler que les petites améliorations au niveau des solutions doivent être visibles à l'algorithme.
- **La taille de la population :** Ce paramètre a une influence sur le comportement de l'algorithme. En effet, une petite population ne crée pas assez d'interactions garantissant le bon fonctionnement de l'algorithme. Cependant, la solution n'est pas simplement d'augmenter le nombre de particules.
- **Les coefficients d'accélération :** Ils prennent souvent la même valeur. Si φ_i est trop petit, alors l'algorithme explorera très lentement, ce qui dégrade sa performance. Cependant, l'expérience a démontré que la valeur de φ_i peut être fixée à 2.05. [Angeline, 98].
- **Le coefficient de resserrement :** Il dépend des φ_i. Si $\varphi_1 = \varphi_2 = 2.05$, alors $k \approx 0.729$.
- **La vitesse maximale :** Avec l'introduction du coefficient de resserrement, l'utilisation de V_{max} n'est pas obligatoire. Cependant, quelques chercheurs annoncent de meilleurs résultats en l'utilisant. En effet, ils suivent une règle simple pour le choix de V_{max} : il s'agit des bornes inférieure et supérieure suivant chaque dimension de l'espace de recherche.
- **Le meilleur voisinage :** c'est un aspect de l'algorithme qui est le moins étudié dans la littérature. Dans la première présentation de l'algorithme, le meilleur voisinage est le même pour toute la population.

2.4.11 L'étude de convergence

Une étude théorique a été élaborée par Ozcan et Mohan [Ozcan et al., 98] sur des modèles simplifiés de l'OEP (ne considérant pas le facteur d'inertie) contenant une seule particule de dimension une. De plus, les paramètres φ_1 et φ_2 sont supposés constants. Les trajectoires de la particule et la convergence de l'essaim sont analysées sous ces conditions.

Ils ont démontré que lorsque $0 < \varphi < 4$, avec $\varphi = \varphi_1 + \varphi_2$, la trajectoire de la particule est une courbe sinusoïdale dont l'amplitude et la fréquence sont fonctions du choix des paramètres et des conditions initiales (la position initiale et la vitesse initiale).

D'autres travaux ont été effectués par Clerc et Kennedy [Clerc et al., 02] qui ont démontré la convergence de l'OEP en utilisant le coefficient de resserrement K.

Cependant, ces différents travaux ne démontrent pas la convergence de l'OEP vers un optimum global ou un optimum local: la convergence est assurée uniquement vers la meilleure position visitée par tout l'essaim.

Dans ce cadre, van den Bergh [van den Berg, 02] a démontré que l'OEP classique ne peut être considéré ni comme un algorithme d'optimisation locale ni comme un algorithme d'optimisation globale. Pour le rendre compétitif dans le cadre de l'optimisation globale, il suggère de lui ajouter deux mécanismes:

- L'introduction d'un opérateur de mutation qui est généralement appliqué sur les guides ou les meilleures positions de l'essaim : pour permettre à l'essaim de s'échapper des états de stagnation éventuels.
- La réinitialisation de l'essaim lorsque l'algorithme converge vers un état d'équilibre : pour permettre à l'essaim de visiter d'autres zones de l'espace de recherche.

2.5 L'optimisation par Essaim Particulaire et l'optimisation multiobjectif

Pour appliquer l'optimisation par Essaim Particulaire sur les problèmes multiobjectif, il est clair que le schéma global de l'OEP classique doit être modifié [Sierra et al., 06]. On doit essentiellement:

- maximiser le nombre d'éléments trouvés qui appartiennent à l'ensemble de Pareto optimal,
- minimiser la distance entre le front de Pareto produit et le front de Pareto réel,
- maintenir la diversité au sein de l'ensemble trouvé.

Pour ce faire, plusieurs problèmes se posent et concernent particulièrement :

- le choix des guides (personnels et globaux) tout en favorisant le choix des particules non dominées par rapport aux particules dominées,
- la manière de retenir les particules non dominées pour qu'elles soient non seulement non dominées vis-à-vis de la population courante mais aussi vis-à-vis des populations passées,
- le maintien de la diversité au sein de l'essaim.

Nous présentons par la suite quelques solutions proposées, dans la littérature, pour traiter ces problèmes. Nous terminons par une présentation des principales techniques d'OEP multiobjectif.

2.5.1 La sélection et la mise à jour des guides

Le choix des guides représente une étape cruciale qui détermine la qualité de l'algorithme de l'OEP multiobjectif produit. Une approche classique est de considérer toutes les particules non dominées comme des candidates potentielles. On peut donc choisir arbitrairement une particule parmi celles non dominées. Dans la littérature, plusieurs approches ont considéré d'autres critères supplémentaires pour choisir le guide. Ces critères à savoir le *sharing* et le *crowding* favorisent la diversité.

2.5.1.1 Le *sharing*

Le *sharing* consiste à modifier la valeur du coût d'une particule (calculé uniquement à partir de la fonction objectif du problème). C'est cette nouvelle valeur qui sera utilisée comme valeur d'adaptation par l'opérateur de sélection du guide. Pour éviter qu'un trop grand nombre de particules ne se concentrent autour d'un même point, il faut pénaliser la valeur d'adaptation en fonction du nombre de particules au voisinage du regroupement : plus les particules sont regroupées, plus leur valeur d'adaptation est faible, et des particules proches les unes des autres doivent partager leur valeur d'adaptation. Dans la pratique, on estime ce taux de concentration en ouvrant un domaine autour d'une particule, puis on calcule les distances entre les particules contenues dans ce domaine.

Pour déterminer les bornes du domaine ouvert autour de la particule choisie, on définit une distance maximale, appelée σ_{share}. Au delà de celle-ci les particules ne seront plus considérées comme faisant partie du domaine ouvert.

2.5.1.2 Le *crowding*

L'approche par *crowding* consiste à déterminer un représentant par niche découverte. A la différence du *sharing*, où toutes les particules sont susceptibles d'être sélectionnées, avec le *crowding*, seuls les représentants peuvent être sélectionnés. Le principe de cette technique consiste à remplacer des éléments d'une population par d'autres éléments similaires et meilleurs qu'eux. En effet, le *crowding* permet de conserver les différentes niches de la population tout en accélérant la convergence.

2.5.2 L'archivage des particules non dominées

La manière la plus répandue pour sauvegarder les particules non dominées est d'utiliser une archive externe. Une particule est ajoutée à l'archive quand elle n'est pas dominée par celles de l'archive. De plus, les particules de l'archive qui sont dominées doivent être supprimées. Cette approche a l'inconvénient d'être coûteuse de point de vue complexité. En effet, la taille de l'archive croît rapidement : le temps de calcul nécessaire pour la mise à jour de l'archive peut s'avérer très coûteux. Pour remédier à ce problème, il est commun de définir une taille maximale pour l'archive. De ce fait, il s'avère nécessaire d'ajouter des règles d'inclusion à l'archive dans le cas où celle-ci est pleine.

Pour l'OEP multiobjectif, théoriquement, il faut utiliser deux archives : une première pour sauvegarder les guides globaux et une deuxième pour sauvegarder, pour chaque particule, les positions non dominées visitées.

2.5.3 Les méthodes existantes

Nous présentons une liste non exhaustive des travaux effectués dans ce domaine.

2.5.3.1 Les méthodes agrégées

Parsopoulos et Vrahatis [Parsopoulos et al., 02] proposent trois types d'agrégation différents : une agrégation linéaire classique, pour laquelle les poids sont fixés, une agrégation dynamique (les poids sont modifiés graduellement au cours du traitement), une agrégation dont les poids sont modifiés brutalement au cours du traitement.

2.5.3.2 Les méthodes non agrégées, non Pareto

2.5.3.2.1 La méthode lexicographique

Hu et Eberhart [Hu et al., 02] proposent un algorithme optimisant un seul objectif à la fois en utilisant un ordre lexicographique. Ils présentent une approche basée sur une génération dynamique des voisinages pour chaque particule. En effet, chaque particule calcule la liste des distances qui la séparent des autres particules de l'essaim. Le voisinage est ainsi formé par les *m* particules les plus proches. La meilleure particule de ce voisinage sera considérée comme son guide local.

Le problème est de définir la métrique de distance, le cardinal d'un voisinage et le moyen de choisir le guide local. L'auteur considère le cas d'un problème à deux objectifs et il propose la démarche suivante:

- fixer le coût suivant le premier objectif ;
- optimiser le deuxième objectif.

Pour générer le voisinage d'une particule donnée, on suit les étapes suivantes:

- calculer les distances qui la séparent des autres particules de l'essaim dans l'espace des objectifs suivant la première fonction objectif ;
- choisir les *m* particules les plus proches en se basant sur cette distance ;
- trouver le guide local en se basant sur les valeurs de la deuxième fonction objectif ;

Concernant le choix du guide personnel (la meilleure position trouvée par une particule), l'auteur adopte une stratégie basée sur la dominance au sens de Pareto qui met à jour le guide personnel dans le cas où la nouvelle position rencontrée le domine.

En pratique, la taille d'un voisinage est fixée à trois (choisir deux autres particules). De plus, il fixe la fonction objectif la plus simple à calculer et il optimise la fonction la plus difficile.

Par la suite, les auteurs proposent une amélioration pour remédier au problème suivant. En effet, ils considèrent le cas de la figure 2.4.

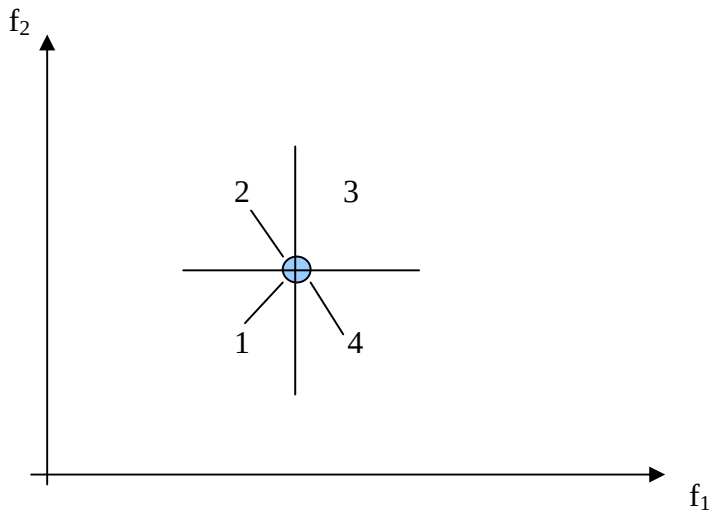


Figure 2.4 : Les directions possibles de déplacement pour une particule

Cette figure présente les directions possibles de déplacement pour une particule donnée. En effet, elle domine toutes les particules de la région 3. Cependant, elle est dominée par toutes les particules de la région 1. De plus, les particules de la région 2 et de la région 4 ne la dominent pas. Néanmoins, ces deux régions présentent des candidates potentielles pour être des solutions Pareto optimales. En fait, ces régions ne dominent pas la particule actuelle mais peuvent dominer d'autres particules.

Dans le DNPSO original, la meilleure performance n'est mise à jour que si la position trouvée domine l'ancienne position (déplacement uniquement dans la région 1). Pour remédier à ce problème, les auteurs introduisent une archive externe pour sauvegarder les positions Pareto optimales. La sélection du guide est similaire à celle considérée dans le DNPSO mais cette fois la sélection est faite parmi les éléments de l'archive externe.

2.5.3.2.2 La méthode VEPSO

Cette stratégie présente une adaptation de VEGA à l'optimisation par essaim particulaire. En effet, on utilise M (avec $M \geq 2$) essaims différents pour explorer les différentes zones de l'espace de recherche. Chaque essaim est évalué exclusivement suivant un objectif. Mais, son mouvement est influencé par l'information arrivant des autres essaims. Cet échange peut conduire l'essaim vers des points Pareto optimaux.

On part d'un essaim formé par M sous essaims. Chaque sous essaim i est évalué suivant un objectif j: on définit pour chaque essaim la meilleure performance des particules suivant cet objectif, que l'on note g_j. En fait, chaque particule i de l'essaim se déplace en se basant sur sa meilleure performance suivant l'objectif j et sur la meilleure performance g_l d'un autre essaim l avec $l \neq j$ (il est choisi arbitrairement).

Cette approche a été validée expérimentalement sur des fonctions bi-objectif connues de la littérature et présentant différentes difficultés comme la convexité, la non convexité, la discontinuité.

Les expérimentations sont faites en faisant varier le nombre de sous essaims à savoir 2, 4, 6, 8 et 10. Ces résultats ont été comparés à ceux de VEGA. Les métriques utilisées pour la comparaison sont la covariance et la largeur du front. En effet, les résultats trouvés ont démontré que VEPSO est plus performant que VEGA pour toutes les fonctions considérées. De plus, l'augmentation du nombre de sous essaims n'a pas d'effet significatif sur la qualité des résultats trouvés [Parsopoulos et al., 04]. Cette méthode est facilement implémentable et aussi parallélisable. Son inconvénient majeur est la favorisation des particules qui sont

performantes suivant les objectifs et l'ignorance des particules moyennes. Ce qui crée le phénomène de spécialisation des particules suivant chaque objectif.

2.5.3.3 Les méthodes Pareto

2.5.3.3.1 Les méthodes non élitistes

- Moore et Chapman [Moore et al., 99] proposent un algorithme utilisant la dominance de Pareto et l'OEP avec une topologie de voisinage circulaire. Dans cette approche, le choix du guide personnel, pour chaque particule, se fait à partir d'une liste contenant les positions non dominées trouvées: le choix du guide se fait arbitrairement à partir de cette liste. La meilleure particule du voisinage, les listes des guides personnels des particules du voisinage sont comparées.
- Ray et Liew [Ray et al., 02] proposent un algorithme d'OEP utilisant la dominance de Pareto. Ils combinent les techniques évolutionnaires avec celles des essaims particulaires. De plus, ils utilisent un opérateur de densité sur le voisinage pour promouvoir la densité dans l'essaim.

2.5.3.3.2 Les méthodes élitistes

a) La technique de Coello

Cette approche [Coello et al., 02] est basée sur l'idée d'avoir une archive externe pour sauvegarder les positions non dominées. De plus, les mises à jour de l'archive sont exécutées en considérant un système géographique qui décompose l'espace des objectifs en un ensemble d'hypercubes. Chaque hypercube est affecté du nombre de particules qui s'y trouvent. L'archive est utilisée également par les particules pour identifier un leader qui guidera la recherche.

b) La technique DOPS

L'auteur propose une technique d'OEP, appelée DOPS, pour l'optimisation multiobjectif. DOPS a intégré des techniques d'archivage bien connues dans les algorithmes évolutionnaires. L'intégration de ces techniques dans l'OEP et leurs extensions sont empiriquement analysées en utilisant plusieurs fonctions de test pour illustrer la rentabilité de l'approche proposée [Bartz-Beielstein et al., 03].

Plusieurs techniques pour la sélection des leaders et la mise à jour des archives ont été employées. En effet, chaque particule est affectée d'une valeur de sélection, résumant l'aptitude de la particule à être sélectionnée à partir de l'archive, et une valeur de suppression,

résumant l'aptitude de la particule à être supprimée de l'archive. Ces deux valeurs sont trouvées suite à l'application des deux fonctions f_{sel} et f_{del}: la fonction de sélection f_{sel} est définie comme étant une mesure de l'influence de chaque particule sur la diversité du front de Pareto. Chaque fois qu'une position leader est nécessaire, un membre de l'archive est choisi selon f_{sel}. Dans le cas où le nombre de positions non dominées trouvées surpasse la taille de l'archive, un membre de l'archive est choisi pour la suppression selon f_{del}.

Les fonctions f_{del} utilisées sont:

+ **Grille adaptative (0):** cette fonction utilise une grille adaptative qui est redimensionnée à chaque génération. Cette grille décompose l'espace des objectifs en un ensemble d'hypercubes. f_{del} est définie comme étant le nombre de particules dans le même hypercube. f_{del} pénalise les particules qui forment des clusters de particules de grandes tailles en terme de nombre.

+ **Distance métrique (1):** cette fonction utilise une distance métrique qui est basée sur les distances relatives dans les archives. Le problème de cette approche est son grand coût en terme de complexité.

+ **Distance randomisée (2):** C'est une variation randomisée de type 1.

Les fonctions f_{sel} utilisées sont:

+ **Sélection uniforme :** toutes les particules de l'archive ont la même probabilité de sélection.

+ **Sélection anti-groupement :** cette technique est basée sur la fonction de suppression. Elle empêche les particules appartenant à des groupes de grandes tailles d'être choisies comme des leaders.

+ **Sélection à base de succès :** l'idée derrière cette fonction est de récompenser les membres de l'archive qui mènent souvent à de nouvelles particules non dominées et ceci en les favorisant dans le processus de sélection.

Les expérimentations sont faites en faisant varier la taille de l'essaim, le facteur d'inertie, la taille de l'archive, les fonctions f_{sel} et f_{del}. En effet, il a été démontré, expérimentalement, que la taille de l'essaim est un facteur important qui détermine la qualité des solutions trouvées, contrairement au facteur d'inertie dont l'effet est relativement insignifiant. Concernant la taille de l'archive, les valeurs modérées par rapport à la taille de l'essaim donnent généralement les meilleurs résultats. De plus, la variation au niveau de f_{sel} n'a pas

d'effet sur la qualité des solutions. C'est pourquoi, il est recommandé d'utiliser la sélection uniforme (la plus simple à mettre en œuvre). Cependant, la fonction f_{del} a un effet plus important: les fonctions (1) et (2) sont meilleures que la fonction (0).

c) La technique de Pulido

L'approche proposée par Pulido et Coelho consiste à utiliser plusieurs essaims (chacun avec une taille fixe). Chaque essaim survolera une région spécifique de l'ensemble de Pareto optimal et aura ses propres guides. Comme la sélection appropriée des leaders est essentielle pour assurer la bonne performance de l'OEP dans les problèmes multiobjectif, les auteurs ont proposé quelques stratégies à cet effet et qui sont les suivantes : (1) Aléatoirement (un leader est aléatoirement choisi) ; (2) le plus proche (une particule choisit comme un leader celui qui est géographiquement le plus proche) ; (3) un seul leader est choisi par toutes les particules.

Chaque essaim fonctionne d'une manière autonome : une fois que tous les essaims ont fini leurs mouvements, l'ensemble des leaders est décomposé en un ensemble de *clusters* en se basant sur le critère de proximité. Chaque groupe résultant sera affecté à un essaim. Ces guides de particule essayeront de surpasser chaque essaim dans l'itération suivante [Pulido et al., 04].

Le pseudo-code de l'algorithme est donné par l'algorithme 4.

Algorithme 4 : Technique de G.T. Pulido

Pour chaque essaim
  Initialiser les particules
  Initialiser g_{leader} (l'ensemble global des leaders)
Fin Pour
Répéter
  Pour chaque essaim
    Répéter
      Pour chaque particule
        Sélectionner le leader
        Effectuer le déplacement
        Mettre à jour l'ensemble de g_{leader}
      Fin Pour
    Jusqu'à atteindre s_{gmax}
  Fin Pour
  Rassembler les g_{leader} dans un seul ensemble
  Appliquer une stratégie de *clustering* pour le diviser en $n_{essaims}$ groupes
  Affecter chaque groupe à chaque essaim
Jusqu'à atteindre G_{max}

L'exécution de cet algorithme peut être divisée en trois phases: l'initialisation, les mouvements et la génération des résultats. En effet, on commence par l'initialisation de chaque essaim (initialisation des particules et de l'ensemble des leaders). Par la suite, chacun effectue ses déplacements. Une fois que tous les mouvements sont réalisés, les leaders de tous les essaims sont regroupés dans un seul ensemble qui est divisé par la suite en $n_{essaims}$ groupes par le biais d'un algorithme de *clustering*.

Les paramètres de cet algorithme sont les suivants:

- G_{max}: le nombre maximal de générations,
- $n_{particules}$: le nombre total de particules,
- $n_{essaims}$: le nombre d'essaims,
- s_{gmax}: le nombre maximal d'itérations pour chaque essaim.

d) L'approche par hybridation de l'OEP avec des algorithmes de recherche locale

Les auteurs proposent une hybridation de l'optimisation des essaims particulaires avec des méthodes de recherche locale à savoir la recherche par dispersion. En effet, l'approche proposée est décomposée en deux phases. Chacune d'entre elles consume un nombre fixé d'évaluations de la fonction objectif. Dans la première phase, on applique l'optimisation par essaim particulaire pour un nombre maximal d'évaluations égal à 2000. Une recherche locale est appliquée dans la deuxième phase pour un nombre maximal d'évaluations égal à 2000.

Dans la première phase, la taille de la population considérée est très petite (P=5 particules). Le leader est déterminé en utilisant le critère suivant: les k premières particules (k étant le nombre d'objectifs) sont guidées par les meilleures particules suivant chaque objectif. Les (P-k) particules restantes sont utilisées pour construire une approximation du vecteur idéal. En fait, on choisit la particule dont le coût est le plus proche du vecteur idéal. Cette particule sera considérée comme un guide pour les (P-k) dernières particules. Le but de la première stratégie de sélection est d'approcher l'optimum suivant chaque objectif en utilisant le taux de convergence élevé dans le cadre de l'optimisation mono-objectif. La deuxième stratégie de sélection est utilisée pour promouvoir la convergence vers le front de Pareto. Les particules Pareto optimales sont sauvegardées dans une archive externe qui est mise à jour en utilisant le principe de l'ε-dominance.

Dans la deuxième phase, on applique une recherche locale dont le but est d'améliorer la qualité des solutions trouvées. Les algorithmes considérés à savoir la recherche par dispersion prennent comme entrée deux ensembles: l'archive externe et l'ensemble des particules dominées qui sont trouvées dans la première phase [Quintero et al., 08].

e) L'approche SigmaMOPSO

Dans cet algorithme [Mostaghim et al., 03], une valeur sigma est affectée à chaque particule de l'essaim et à chaque particule de l'archive externe. Alors, une particule donnée de l'essaim choisit comme son leader la particule de l'archive externe qui a la plus proche valeur de sigma. L'utilisation des valeurs de sigma peut causer la convergence prématurée dans quelques cas. Pour éviter ceci, un opérateur de turbulence est utilisé, qui est appliqué sur l'espace des variables de décision.

f) L'approche MOPSO-pd

Dans cet algorithme [Alvarez-Benitez et al., 05], les auteurs proposent des méthodes basées exclusivement sur la dominance Pareto pour choisir les leaders à partir de l'archive contenant les particules non-dominées. Trois techniques de sélection différentes sont présentées : une technique qui favorise explicitement la diversité, une technique qui promeut explicitement la convergence et finalement une technique qui est une méthode pondérée cherchant un compromis entre la diversité et la convergence. De plus, MOPSO-pd utilise un facteur de turbulence pour promouvoir la diversité dans l'essaim.

g) L'approche de Reyes-Sierra (OMOPSO)

L'algorithme proposé est basé sur la dominance de Pareto : chaque position non dominée présente une candidate potentielle pour être sélectionnée comme étant un leader. Une fonction d'encombrement est également utilisée pour filtrer l'ensemble des leaders. En effet, l'essaim est divisé en trois sous essaims de même taille : dans chaque sous essaim, on applique une technique de mutation différente. En fait, la mutation est utilisée pour promouvoir la diversité dans l'essaim.

Pour chaque particule, le leader est choisi arbitrairement dans l'ensemble des leaders existants avec une probabilité P_s. Sinon il est sélectionné avec une probabilité $(1-P_s)$ en se basant sur la fonction d'encombrement. De plus, l'ensemble des leaders est sauvegardé dans une archive externe dont la mise à jour est basée sur la fonction d'encombrement et ceci pour favoriser le maintien des individus se situant dans les zones les moins encombrées. Cette approche intègre aussi le concept de l'ε-dominance pour fixer la taille de l'archive [Sierra et al., 05].

Les déplacements sont effectués en respectant les équations suivantes:

$$v_{id} = \boldsymbol{w} v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \qquad (2.1)$$

$x_{id} = x_{id} + v_{id}$ (2.2)

Le pseudo-code de l'algorithme est donné par l'algorithme 5.

Algorithme 5 : MOPSO

Initialiser aléatoirement la population et initialiser l' ε-archive
Calculer la valeur de la fonction d'encombrement pour chaque leader
Répéter
  Pour chaque particule
    Choisir le leader
    Effectuer le déplacement
    Effectuer la mutation
    Evaluer la particule
    Mettre à jour la meilleure performance de la particule
  FinPour
  Mettre à jour l'ε-archive
  Mettre à jour les valeurs de la fonction d'encombrement pour chaque leader
Jusqu'à Nombre Maximal d'itérations atteint

Les paramètres qui interviennent sont:

- la taille de l'essaim,
- la probabilité P_s pour choisir le leader,
- les paramètres c_1 et c_2,
- le facteur d'inertie *w*,
- le nombre maximal d'itérations.

L'auteur a effectué une étude permettant de définir l'impact de chaque paramètre sur la performance de son approche. Il a considéré plusieurs combinaisons possibles en faisant varier les valeurs pour chaque paramètre. Le critère choisi pour comparer les différentes variantes de son approche est le taux de succès.

Les résultats trouvés sont les suivants:

- Pour la taille de l'essaim et le nombre maximal d'itérations, les grandes valeurs donnent les meilleurs résultats.
- Pour les paramètres P_s, c_2 et *w* les grandes valeurs donnent les meilleurs résultats.
- Le paramètre c_1 n'a pas d'effet sur la performance de l'algorithme.

Les meilleures valeurs pour chaque paramètre sont les suivantes:

- Pour P_s: entre 0,8 et 0,97,
- Pour C_2: entre 1,9 et 2,
- Pour *w*: entre 0,4 et 0,5.

Introduction de l'héritage et de l'approximation dans l'évaluation d'un individu

Vu le coût et la complexité de l'évaluation d'une particule suivant les fonctions objectifs, l'auteur a proposé l'utilisation des mécanismes d'héritage et d'approximation pour l'évaluer et ceci afin de réduire la complexité tout en garantissant le fait que le coût trouvé est fiable. Notons que ces particules ne seront pas introduites dans l'archive car leurs coûts sont fictifs.

L'algorithme devient ainsi [Sierra et al., 07] (voir Algorithme 6) :

Algorithme 6 : MOPSO avec approximation de l'évaluation

Initialiser aléatoirement la population et initialiser l' ε-archive
Calculer la valeur de la fonction d'encombrement pour chaque leader
Répéter
  Pour chaque particule
    Choisir le leader
    Effectuer le déplacement
    Effectuer la mutation
    Si (Pr_i)
      Hériter
    Sinon
      Evaluer la particule
    Fin Si
    Mettre à jour la meilleure performance de la particule
  FinPour
  Mettre à jour l'ε-archive
  Mettre à jour les valeurs de la fonction d'encombrement pour chaque leader
Jusqu'à critère d'arrêt vérifié

Les techniques d'héritage utilisées pour chaque particule sont:

- des combinaisons linéaires des coûts du leader, de la meilleure performance personnelle et l'ancien coût. Ces combinaisons sont faites de plusieurs manières en négligeant à chaque fois l'un des facteurs de cette combinaison.
- Application des formules de déplacement dans l'espace des objectifs.

Les techniques d'approximation utilisées pour chaque particule sont:

- En considérant l'archive, choisir le leader le plus proche de la particule. Son coût lui sera affecté.
- En considérant la totalité de l'essaim, choisir la particule la plus proche de la particule considérée.
- En considérant la totalité de l'essaim, choisir les dix particules les plus proches de la particule considérée. Son coût sera la moyenne des coûts des particules choisies.

Dans le but de comparer les différentes techniques proposées, l'auteur a réalisé une étude en considérant des problèmes présentant différentes propriétés à savoir la convexité, la non convexité et la discontinuité. Pour faire la comparaison, l'auteur utilise la métrique du taux de succès.

Les résultats trouvés sont les suivants:

- Les meilleures techniques d'héritage sont celles qui ignorent l'ancien coût d'une particule donnée.
- Les meilleures techniques d'approximation sont celles qui considèrent la totalité de l'essaim et non l'archive.
- Les techniques d'héritage sont les plus performantes dans le cas d'un espace de décision avec une grande dimension. Dans le cas contraire (les petites dimensions), les techniques d'approximation sont les plus performantes.

h) La technique de Durillo

Les auteurs ont présenté dans [Durillo et al., 09] le *Speed-constrained* MOPSO (SMPSO). Ils présentent un mécanisme pour borner la vitesse dans l'algorithme OMOPSO afin d'améliorer la capacité de recherche de l'algorithme. En effet, suite à une étude expérimentale du comportement d'OMOPSO, les auteurs ont constaté que la mauvaise performance peut être justifiée par les valeurs de vitesse de la majorité des particules qui subissent une sorte de comportement irrégulier dans quelques points de l'exécution, alternant des valeurs très élevées avec des valeurs très basses. La conséquence est que ces particules se déplacent à leurs valeurs extrêmes continuellement, donc elles ne contribuent pas à la recherche. Les auteurs ont alors proposé le mécanisme suivant pour borner la vitesse :

$$v_{i,j} = \begin{cases} delta_j & v_{i,j} > delta_j \\ -delta_j & v_{i,j} \leq -delta_j \\ v_{i,j} & sinon \end{cases}$$

Avec $delta_j = \frac{x_{max} - x_{min}}{2}$.

Les résultats expérimentaux ont montré que les mouvements irréguliers de la vitesse ont disparu, donc la particule prend des valeurs à l'intérieur de l'intervalle de la variable et ainsi elle se déplace le long des régions différentes de l'espace de recherche.

i) La technique de Cooren

L'auteur a développé une version multiobjectif de TRIBES. En effet, MO-TRIBES utilise une approche basée sur la dominance au sens de Pareto. Les particules non dominées sont

stockées dans une archive externe dont la taille et les mises à jour sont définies automatiquement. De plus, la diversité est maintenue grâce à un critère de maximisation de distance d'encombrement et aussi grâce aux réinitialisations multiples de l'essaim. Les résultats de MO-TRIBES sont très encourageants [Cooren, 09].

- La mise à jour de l'archive consiste à ajouter toutes les particules non dominées à l'archive et à supprimer les particules dominées déjà présentes. Si le nombre de particules dans l'archive excède un nombre fixé au préalable, on applique une fonction d'encombrement pour réduire la taille de l'archive et aussi maintenir sa diversité. En effet, la diversité des solutions non dominées stockées est maintenue à l'aide d'un critère basé sur la distance d'encombrement. La distance d'encombrement d'un élément donné approche le volume du plus grand hypercube contenant cet élément sans en contenir d'autres. L'idée est de maximiser la distance d'encombrement afin d'obtenir une répartition des particules la plus uniforme possible le long du front de Pareto.
- Vu la nature de TRIBES qui est un OEP sans paramètres, l'auteur définit une règle empirique permettant de définir la taille de l'archive en fonction du problème.
- Le choix des informatrices d'une particule donnée est similaire au cas de TRIBES mono-objectif. En effet, l'informatrice cognitive d'une particule n'est mise à jour que dans le cas où la nouvelle position la domine. Concernant l'informatrice sociale, elle dépend du statut de la particule considérée dans sa tribu: si elle n'est pas la meilleure de sa tribu, son guide est alors la meilleure particule de la tribu. Si l'on considère, en revanche, la meilleure particule d'une tribu donnée, l'informatrice est alors une particule quelconque de l'archive.
- En considérant que l'essaim converge au bout d'un faible nombre d'itérations vers un état de quasi-équilibre, on peut considérer qu'il est inutile de continuer la recherche et qu'il est préférable de recommencer une nouvelle recherche.

k) Le MOPSO-adaptatif

Cette approche consiste à présenter une version adaptative du MOPSO classique. Elle repose sur les DoE (*Designs of Experimentation*) qui permettent de détecter les changements de performance d'un algorithme en faisant varier les valeurs de ses paramètres. Ils permettent aussi de découvrir les effets des différentes interactions entre les différents paramètres.

Pour le cas du MOPSO classique, l'équation de déplacement des particules met en jeu plusieurs paramètres. Dans ce cas les paramètres de contrôle qui influencent la performance

de l'algorithme sont : w, c_1 *et* c_2. Dans chaque génération, deux valeurs différentes de chaque paramètre sont appliquées, conduisant ainsi à 2^3=8 combinaisons différentes. Chacune de ces combinaisons est appliquée à un huitième des membres de la population qui sont aléatoirement choisis [Zielinski et al., 07].

l) Le Micro-MOPSO

Cette approche consiste à utiliser une population de petite taille. La sélection du leader se fait en se basant sur la dominance de Pareto et elle utilise un estimateur de densité afin de promouvoir la diversité. De plus, l'algorithme intègre un processus de réinitialisation et deux archives externes : la première pour stocker les solutions trouvées au cours du processus de recherche et la deuxième pour stocker les solutions finales. En outre, un opérateur de mutation est incorporé pour améliorer les capacités d'exploration de l'algorithme [Cabrera et al., 10].

2.5.4 Problèmes de l'OEP multiobjectif

Le premier inconvénient majeur est la convergence rapide qui est l'une des caractéristiques les plus importantes de l'OEP multiobjectif. En effet, la convergence est assurée uniquement vers les meilleures positions visitées par tout l'essaim. Cette convergence prématurée est provoquée par la perte rapide de diversité dans l'essaim. Ainsi, le maintien de la diversité dans l'OEP est un point très important afin de contrôler sa convergence (normalement rapide). Généralement, lorsqu'un essaim stagne, c'est-à-dire, quand les vitesses des particules sont pratiquement nulles, il devient incapable de produire de nouvelles solutions qui pourraient mener l'essaim hors de cet état. Ce comportement peut mener l'essaim entier à être emprisonné dans un front local duquel il est impossible de s'échapper.

Le deuxième inconvénient majeur de l'OEP multiobjectif est de comporter un nombre élevé de paramètres. En effet, les performances de la méthode face à un problème donné étant fortement liées aux valeurs de ses paramètres de réglage, il est souvent difficile et long de trouver les valeurs optimales de chacun des paramètres.

2.6 Conclusion

Nous avons présenté dans ce chapitre quelques méthodes d'optimisation multiobjectif issues des métaheuristiques pour l'optimisation multiobjectif. Ces méthodes ont montré leur efficacité pour trouver des solutions approchées satisfaisantes pour un large panel de problèmes, sans pour autant que l'utilisateur ait à modifier leur structure. Un intérêt particulier a été porté à la méthode d'optimisation par essaim particulaire. Cette jeune méthode, inspirée

des déplacements d’animaux en essaims, a rencontré un vif succès depuis sa création. Sa relative simplicité et son efficacité en font un des algorithmes les plus utilisés de nos jours. De nombreux axes de recherches ont pour objet de tirer le meilleur parti du paradigme de l’OEP. Parmi ceux-ci, la réduction des paramètres apparaît comme un enjeu crucial, en vue de l’introduction massive de l’OEP dans l’industrie. Les chapitres suivants présentent une méthode d’optimisation particulaire adaptative, qui présente la particularité de ne comporter aucun paramètre de contrôle.

Chapitre 3 Adaptation de TRIBES à l’optimisation multiobjectif

3.1 Introduction

Comme nous l'avons vu dans le chapitre précédent, la famille des métaheuristiques a été utilisée efficacement pour la résolution des problèmes d'optimisation multiobjectif. Elles présentent actuellement un sujet de recherche privilégié. En effet, elles ont été intensément exploitées et elles ont atteint aujourd'hui leurs limites. De ce fait, de nouveaux axes de recherche sont explorés afin de dépasser les restrictions imposées par les algorithmes tels qu'ils ont été initialement définis. Ces nouveaux axes de recherche visent à faciliter la prise en main des métaheuristiques.

Parmi les améliorations possibles, la réduction du nombre de paramètres des algorithmes apparaît comme un enjeu majeur. En effet, la résolution d'un problème donné passe toujours par une étape primordiale qui définit un jeu de paramètres optimal. Cette étape nécessite une connaissance approfondie des mécanismes de l'algorithme utilisé. D'où le succès modéré de ces techniques dans le monde de l'industrie où seules la performance, l'efficacité et la rapidité comptent.

Dans ce chapitre, nous présentons en détail TRIBES, un algorithme d'optimisation par essaim particulaire sans paramètres de contrôle [Clerc, 03]. Son adaptation pour l'optimisation multiobjectif sera aussi présentée.

3.2 Présentation de TRIBES

Afin de résoudre un problème d'optimisation donné, l'utilisateur doit tout d'abord décrire les données spécifiques à ce problème. En effet, il doit délimiter l'espace de recherche, souvent en précisant pour chaque dimension l'intervalle des valeurs admissibles. Il doit également préciser la fonction objectif et le critère d'arrêt.

Un algorithme sans paramètres de contrôle signifie que l'utilisateur intervient uniquement dans l'étape de description du problème. Il n'a aucun paramètre à définir par la suite. De ce fait, l'algorithme doit être capable de définir ces paramètres. Pour ce faire, il doit incorporer des règles définissant comment, à chaque itération de l'algorithme, l'essaim va évoluer et se comporter, tout en intégrant les informations progressivement recueillies au cours du traitement. Ce point de vue consiste à chercher un compromis entre la performance et la facilité de prise en main de l'algorithme.

Dans ce cadre se situe TRIBES, un algorithme d'OEP sans paramètres de contrôle et conçu par Clerc [Clerc, 06]. En fait, TRIBES est défini comme une boîte noire, pour laquelle

l'utilisateur n'a qu'à spécifier le problème à résoudre. Proposer une OEP adaptative impose de répondre à deux types de questions : Comment la structure de l'essaim évolue au cours du temps ? et Quel comportement doit adopter une particule ? La première question est relative à la définition du nombre de particules de l'essaim et de la topologie de voisinage. La deuxième question est relative à la définition des paramètres w, c_1, c_2 et V_{max}. Deux types d'adaptations sont alors définis : les adaptations structurelles et les adaptations comportementales. Ces deux types d'adaptations sont détaillés dans les sections suivantes.

3.2.1 Adaptations structurelles

3.2.1.1 Structure de l'essaim

Dans TRIBES, l'essaim est divisé en plusieurs sous-essaims appelés tribus. Les tribus sont de tailles différentes, qui évoluent au cours du temps. Le but est d'explorer simultanément plusieurs régions de l'espace de recherche, généralement des optima locaux, avant de prendre une décision globale. Pour ce faire, les tribus doivent être capables d'échanger les informations recueillies tout au long du traitement. La structure de l'essaim est présentée dans la figure 3.1. A cet effet, deux types de communication sont nécessaires : la communication intra-tribu et la communication inter-tribu.

La communication intra-tribu désigne l'échange d'informations entre les particules formant la même tribu. Chaque tribu présente une topologie complètement connectée. En effet, chaque particule est capable de connaître la meilleure et la plus mauvaise position jamais atteintes par la tribu.

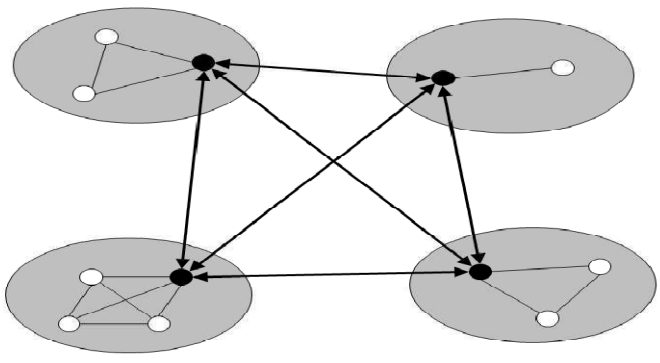


Figure 3.1 : Structure de l'essaim

Tout au long du traitement, chaque tribu va converger vers un optimum local. Pour assurer la diversité dans l'essaim et pour prendre une décision globale, il faut que ces tribus échangent les informations entre elles pour choisir un optimum global parmi les différents

optima locaux. Ce type de communication définit la communication inter-tribu. Cette communication est faite par l'intermédiaire des meilleures particules des tribus.

Chaque particule dispose de trois types d'informateurs dans l'essaim. En effet, elle est informée par elle-même (mémoire cognitive *p*), par tous les éléments de sa tribu (appelés informateurs internes), et, si cette particule est un *chaman* (i.e. la meilleure particule d'une tribu), alors elle est aussi informée par les chamans des autres tribus (appelés informateurs externes).

La mémoire sociale *g* de chaque particule est l'informateur pour lequel la valeur de la fonction objectif est la plus petite.

3.2.1.2 Evolution des tribus

L'évolution des tribus signifie l'ajout ou la suppression de particules. Ceci étant assuré en définissant des règles d'adaptation structurelles pour l'essaim. Pour ce faire, on définit deux indicateurs de qualité, un pour les particules et un pour les tribus. Une particule est dite bonne si elle a amélioré sa meilleure performance au cours de la dernière itération. Dans le cas contraire, elle est dite neutre. Cet indicateur est purement qualitatif, car il est basé sur l'évolution de la performance et non sur la performance elle-même. En plus de cet indicateur, la meilleure et la plus mauvaise position de la tribu sont stockées.

Une tribu est aussi dite bonne ou mauvaise suivant le nombre de bonnes particules présentes en son sein. Une tribu est dite mauvaise si aucune de ses particules n'a amélioré sa meilleure performance au cours de la dernière itération. Les tribus qui possèdent au moins une bonne particule sont déclarées bonnes avec une probabilité de 0,5, mauvaises sinon.

L'utilisation de ces indicateurs permet de définir les règles de création et de destruction de particules.

3.2.1.2.1 Destruction de particules

L'intérêt de la destruction des particules est de minimiser le nombre des particules de l'essaim. Le but de cette opération est de minimiser le nombre d'évaluations de la fonction objectif et de gagner par la suite du temps lors de l'exécution de l'algorithme. Donc, aussitôt qu'une occasion de supprimer une particule avec presqu'aucun risque ne surgit, elle doit être saisie. Notons qu'il vaut mieux conserver une particule par erreur (dans le pire cas, le nombre d'évaluations de la fonction objectif pour atteindre la solution optimale augmentera) plutôt que supprimer une particule par erreur (au risque de complètement manquer la solution optimale).

C’est pourquoi seulement une bonne tribu peut supprimer une de ses particules et seulement la plus mauvaise, parce que les informations qu’elle porte (sa meilleure performance) ne sont probablement pas très utiles. Cependant, dans le cas d’une tribu formée par une seule particule, la suppression aura lieu seulement si l’un de ses informateurs a une meilleure performance. En effet, il faut conserver au moins une particule avec les informations de meilleure qualité que celle supprimée. Sur la Figure 3.2, la tribu T1 est déclarée bonne. La particule P, formant la tribu T1, ne sera détruite que si son informateur externe possède une meilleure performance.

D’autre part, la suppression d’une particule implique une redistribution de ses liens d’information. Dans le cas général, cette redistribution est faite sur la meilleure particule de la tribu. Dans le cas d’une tribu formée par une seule particule, puisque la suppression d’une particule mène à l’effacement de la tribu entière, les liens d’information sont réattribués au meilleur informateur externe de la particule supprimée.

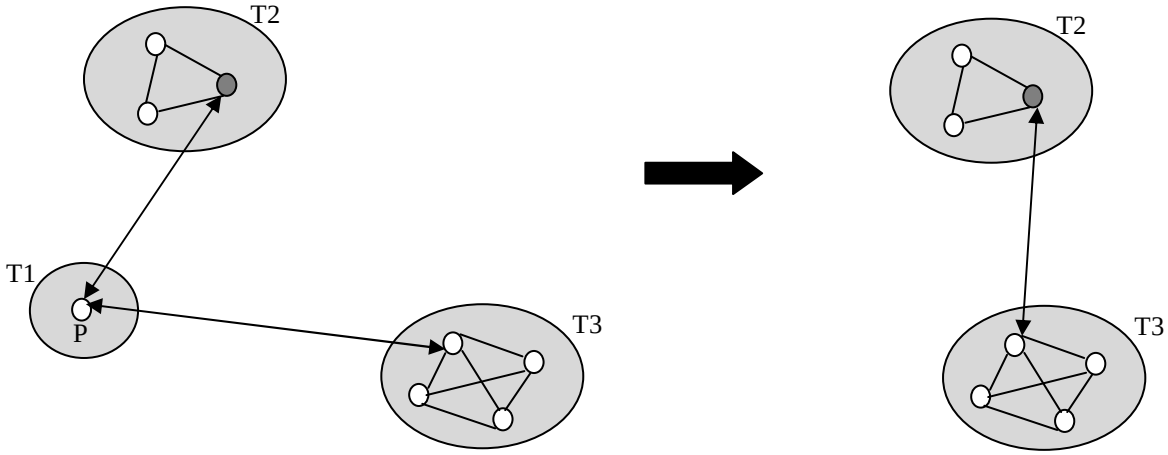


Figure 3.2 : Elimination d’une particule

3.2.1.2.2 Création de particules

Une bonne tribu peut supprimer sa plus mauvaise particule, elle n’aura pas besoin probablement des informations portées par cette particule. En revanche, une mauvaise tribu a besoin de plus d’informations puisqu’aucune de ses particules ne semble converger. De ce fait, la création de nouvelles particules ne concerne que les mauvaises tribus. Chaque mauvaise tribu génère des particules. Le nombre de particules générées par chaque mauvaise tribu est défini par l’équation (3.1). Cette équation est inspirée d’une formule identique prouvée pour l’OEP binaire [Clerc, 06]. Toutes les particules générées par les différentes mauvaises tribus forment une nouvelle tribu.

$$NB_{particules} = \max(2, \left\lfloor \frac{9.5+0.124.(D-1)}{tribeNB} \right\rfloor) \qquad (3.1)$$

Où *D* est la dimension de l'espace de recherche et *tribeNb* le nombre de tribus dans l'essaim.

Le rôle de la nouvelle tribu est d'améliorer les performances des mauvaises tribus. En effet, le chaman de la tribu créée sera l'informateur externe des chamans des mauvaises tribus. Deux types de particules sont générées, le type de particule étant sélectionné aléatoirement :

- Les particules libres :

Ces particules sont générées aléatoirement à l'aide d'une distribution uniforme soit dans tout l'espace de recherche, soit sur une frontière de celui-ci, ou sur un sommet. Les particules sont générées en utilisant (3.2). Le but est d'apporter de la diversité à la population.

Dans tout l'espace de recherche : $X_j = U(x_{min-j}, x_{max-j}), j \in \{1, \dots, D\}$

$$\begin{cases} \text{Sur une frontière :} & X_j = \begin{cases} U(x_{min-j}, x_{max-j}), j \in I \subset \{1, \dots, D\} \\ U(\{x_{min-j}, x_{max-j}\}), j \in J \subset \{1, \dots, D\} \end{cases} \\ \text{Sur un sommet :} & X_j = (\{x_{min-j}, x_{max-j}\}), j \in \{1, \dots, D\} \end{cases} \qquad (3.2)$$

où $U(x_{min-j}, x_{max-j})$ est un réel uniformément choisi dans $[x_{min-j}, x_{max-j}]$ et $U(\{x_{min-j}, x_{max-j}\})$ est un réel uniformément choisi dans la liste$\{x_{min-j}, x_{max-j}\}$. I et J sont deux sous-espaces qui forment une partition de {*1*,…, *D*}. Ces deux espaces sont définis aléatoirement pour chaque particule générée.

- Les particules confinées :

Si *x* est la meilleure particule de la tribu génératrice et si i_x est sa meilleure informatrice, p_x et p_{i_x} sont les meilleures performances de *x* et de i_x. La nouvelle particule est générée dans la D-sphère de centre p_{i_x} et de rayon $\|p_x - p_{i_x}\|$ à l'aide de l'équation (3.3). L'objectif ici est d'intensifier la recherche de la tribu génératrice.

$$X_{generé} = alea_{sphere}(p_{i_x}, \|p_x - p_{i_x}\|) \qquad (3.3)$$

Où $alea_{sphere}(p_{i_x}, \|p_x - p_{i_x}\|)$ est un point uniformément choisi dans une hyper-sphère de centre p_{i_x} et de rayon$\|p_x - p_{i_x}\|$.

3.2.1.3 Fréquences des adaptations

Les adaptations structurelles ne doivent pas être effectuées à chaque itération. En effet, du temps doit être laissé pour que l'information introduite par la dernière modification de la topologie se propage dans l'essaim. Ici de nouveau, plusieurs règles possibles peuvent être formulées pour le réaliser. Théoriquement, le temps nécessaire pour que l'information se propage dans tout l'essaim est égal au diamètre du graphe d'information. A cet effet, on doit considérer toutes les paires possibles de particules appartenant à deux tribus différentes et trouver le plus court chemin les liant, en termes de nombre d'arcs. Le plus long de ces plus courts chemins donnerait une évaluation du nombre d'itérations nécessaires pour être sûr que les informations portées par une particule puissent être transmises, plus ou moins directement à toutes les autres particules. Cependant, si la taille de l'essaim devient trop importante, ce nombre peut vite devenir très long à calculer. En pratique, si NL est le nombre de liens d'informations lors de la dernière adaptation, la prochaine adaptation sera effectuée NL/2 itérations plus tard. NL est estimé à l'aide de l'équation suivante :

$$NL=\sum_{n=1}^{tribeNB} particuleNB[n]^2 + tribeNB(tribeNB-1) \qquad (3.4)$$

Où tribeNb est le nombre de tribus et *particuleNb[n]* est le nombre de particules de la tribu n. Le premier terme du membre de droite de l'équation (3.4) suppose que les particules d'une tribu sont complètement connectées. Donc, pour la tribu *n*, le nombre de liens d'information correspondant à la communication intra-tribu est égal à $particuleNb[n]^2$. Le second terme suppose quant à lui que tous les chamans sont reliés entre eux. Chaque chaman est relié à tous les autres, donc il possède tribeNb-1 liens d'informations correspondant à la communication inter-tribu. Les adaptations structurelles peuvent être résumées par l'Algorithme 7.

3.2.1.4 Evolution de l'essaim

Au début du traitement, l'essaim est composé d'une seule particule formant une seule tribu. Après la première itération, si la situation ne s'améliore pas, ce qui est très probable (même certain avec les stratégies de déplacement adoptées, pas de déplacement significatif surtout pendant les premières itérations), une autre particule sera produite, formant une deuxième tribu.

A l'itération suivante, si aucune des deux particules n'améliore sa situation, chacune des deux tribus produira une nouvelle particule, formant ainsi une nouvelle tribu de deux particules et le processus continuera. Comme le nombre de liens augmente, le nombre d'itérations entre deux adaptations successives augmente. Ainsi quand les choses vont mal, de

plus grandes tribus seront produites, augmentant ainsi la capacité de recherche de l'essaim. Entre deux adaptations, l'essaim a alors de plus en plus de chances de trouver une solution.

Au contraire, aussitôt qu'une solution est trouvée, chaque tribu enlèvera progressivement sa plus mauvaise particule, probablement jusqu'à sa suppression complète. Idéalement, quand la convergence devient de plus en plus certaine, toutes les tribus sauf la dernière produite sont réduites à une ou deux particules. Pratiquement, l'essaim a tendance à progresser de plus en plus lentement.

Algorithme 7 : Adaptations structurelles

```
test=0
Pour i=1 à tribeNb faire
   Si tribe[i].statut=mauvais
      Générer NB_particules = Max(2, ⌊(9.5+0.124.(D−1))/tribeNB⌋)
      test=1
   Fin Si
   Si tribe[i].statut=bonne
      Supprimer la plus mauvaise particule de tribe[i]
      Si tribe[i].taille≠1
         Rediriger les liens d'information vers la meilleure particule de la tribu
      Sinon
         tribeNb=tribeNb-1
         Rediriger les liens d'information vers la meilleure informatrice
      Fin Si
   Fin Si
Fin Pour
Si test=1
   tribeNb= tribeNb+1
   Agréger les particules générées à la nouvelle tribu
   Etablir un lien entre les différents chamans
Fin Si
Calculer NL
```

Tableau 3.1 : Stratégies de déplacement

Historique	Stratégies de déplacements
(=+) (++)	locale par gaussiennes indépendantes
(+=) (-+)	pivot bruité
(--) (=-)(+-) (-=) (==)	pivot

Si, durant la première itération, cette particule n'améliore pas sa performance, de nouvelles particules vont alors être générées pour former une deuxième tribu. NL/2 itérations plus tard, le même procédé est répété. On continue selon ce schéma durant toute l'exécution.

Pendant les premières itérations, l'exploration de l'espace de recherche est favorisée à l'intensification. Quand l'essaim couvre l'espace de recherche et le nombre de ses particules augmente, le temps séparant deux adaptations va être plus long et le volet intensification et exploitation devient plus important et les particules auront ainsi le temps nécessaire pour trouver les bonnes solutions.

Une fois que des zones prometteuses de l'espace de recherche sont détectées, les moins bonnes particules vont être progressivement supprimées.

3.2.2 Adaptations comportementales

Les adaptations comportementales concernent les stratégies de déplacement des particules, en fonction de leur comportement passé. Le but est de permettre aux bonnes particules d'intensifier la recherche dans leur zone, considérée comme prometteuse et, inversement, aux moins bonnes particules de diversifier leur recherche.

Il y a trois possibilités d'évolution pour décrire l'historique d'une particule entre deux itérations : amélioration, détérioration ou statu quo, suivant que la particule a amélioré, détérioré ou égalé sa performance précédente. Ces trois états sont notés en utilisant les symboles suivants : + pour une amélioration, - pour une détérioration et = pour un statu quo. Le passé récent d'une particule est constitué par ses deux dernières variations. Par exemple, une amélioration suivie d'une détérioration est notée (+ -). Il y a donc neuf possibilités de passés possibles. Cependant, on se contente de les diviser en trois groupes (voir Tableau 3.1).
(= +) et (+ +) représentent les particules qui ont le meilleur comportement, donc on choisit d'intensifier la recherche dans leur zone à l'aide d'une stratégie locale à gaussiennes indépendantes. (+ =) et (- +) représentent les particules au comportement moyen. Il leur est attribué une stratégie par pivots bruités. Enfin, (- -), (- =), (+ -), (- =) et (= =) représentent les plus mauvaises particules. On leur attribue donc une possibilité de déplacement relativement large, en utilisant une stratégie par pivot.
Les stratégies de déplacement sont définies de la manière suivante :

- Pivot :

On note *p* la meilleure position jamais atteinte par la particule, *g* sa meilleure informatrice et f la fonction objectif. Le mouvement est déterminé par :

$$x = c_1.alea_{sphere}(H_p) + c_2.alea_{sphere}(H_g) \quad (3.5)$$

Où c1=$\frac{f(p)}{f(p)+f(g)}$, c2=$\frac{f(g)}{f(p)+f(g)}$, $alea_{sphere}(H_p)$ est un point uniformément choisi dans l'hyper-sphère de centre p et de rayon $\|p-g\|$et $alea_{sphere}(H_g)$ est un point uniformément choisi dans l'hyper-sphère de centre g et de rayon $\|p-g\|$.

- Pivot bruité :

Cette stratégie est identique à la précédente, mais un bruit est ajouté.

$$x = c_1.alea_{sphere}(H_p) + c_2.alea_{sphere}(H_g) \quad (3.6)$$

$$b=N\left(0,\frac{f(p)-f(g)}{f(p)+f(g)}\right)$$

$$x=(1+b).x$$

- Locale par gaussiennes indépendantes :

Si g est la meilleure informatrice de la particule, le mouvement est déterminé comme suit :

$$x_j = g_j + alea_{normal}(g_j - x_j, \|g_j - x_j\|), j \in \{1, \dots, D\} \quad (3.7)$$

Où $alea_{normal}(g_j - x_j, \|g_j - x_j\|)$ est un point aléatoirement choisi à l'aide d'une distribution gaussienne de moyenne $g_j - x_j$ et de variance $\|g_j - x_j\|$.

Il est important de noter que, contrairement à l'OEP originale, le concept de vitesse n'apparaît pas dans TRIBES. Il est précisé dans [Clerc, 06] que la suppression de ce concept n'est pas un obstacle à l'OEP. La seule différence notable est que la distribution de probabilité de la prochaine position est modifiée, elle est D-sphérique dans le cas de TRIBES (voir Figure 3.4) et D-parallélépipédique dans le cas de l'OEP classique (voir Figure 3.3) [Clerc, 06]. Ceci implique que, dans TRIBES, le déplacement n'est pas dépendant du système de coordonnées. Le schéma global de l'OEP classique est respecté, le vecteur vitesse pouvant être déduit par *x(t+1) +x(t).*

L'algorithme de TRIBES est donné par l'algorithme 8. g_i est la meilleure informatrice de la particule i, p_i est la meilleure position atteinte par la particule i. n est le nombre d'itérations effectuées depuis la dernière adaptation structurelle.

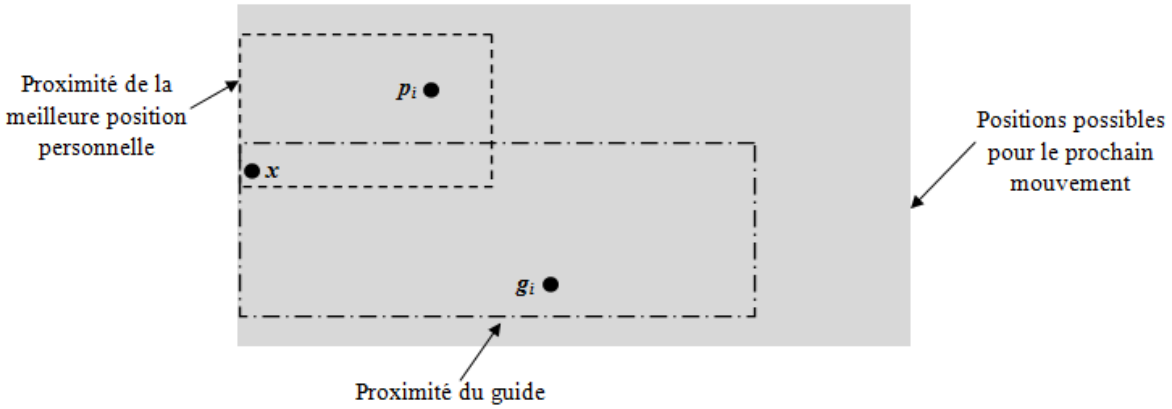


Figure 3.3 : Distribution de la prochaine position dans le cas d'une distribution D-parallélépipédique

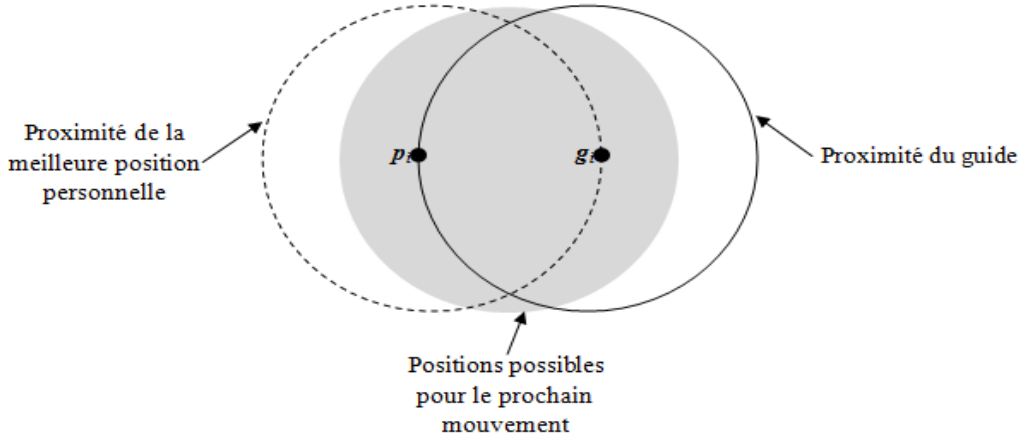


Figure 3.4 : Distribution de la prochaine position dans le cas d'une distribution D-sphérique

3.3 Adaptation de TRIBES à l'optimisation multiobjectif

TRIBES-Multiobjectif a pour objectif d'être un algorithme d'optimisation par essaim particulaire multiobjectif sans paramètres de contrôle. Il reprend les principaux mécanismes de TRIBES auxquels sont ajoutés de nouveaux mécanismes destinés à traiter des problèmes multiobjectif. TRIBES-Multiobjectif a été conçu en utilisant une approche basée sur la dominance au sens de Pareto.

Dans le but d'adapter le paradigme de TRIBES à la résolution de problèmes multiobjectif, il est clair que le schéma général de l'algorithme doit être modifié. En général, la résolution d'un problème multiobjectif pose trois problèmes :

- trouver le plus d'éléments possibles appartenant au front de Pareto ;
- approcher le mieux possible le front de Pareto ;
- la répartition des solutions doit être la plus uniforme possible le long du front de Pareto approché.

Algorithme 8 : Algorithme de TRIBES

Initialisation aléatoire de la particule initiale x_0
$p_0 = g_0 = x_0$
Evaluer la fonction objectif
Tant que le critère d'arrêt n'est pas atteint, **faire**
  Déterminer le statut de chaque particule
  Choisir la stratégie de déplacement
  Déplacement des particules
  Evaluer la fonction objectif
  Mise à jour des p_i et des g_i
  Si n<NL
    Déterminer la qualité des tribus
    Adaptations structurelles
    Calculer NL
  Fin Si
Fin Tant que

Pour adapter TRIBES à l'optimisation multiobjectif, on doit essentiellement garantir l'obtention d'un ensemble de solutions proches du front du Pareto réel, tout en maintenant la diversité au sein de cet ensemble. Pour ce faire, plusieurs problèmes se posent et concernent particulièrement :

- le choix des particules « informatrices » pour chaque tribu,
- le choix de la meilleure performance pour chaque particule,
- le maintien des particules non-dominées rencontrées au cours de la recherche,
- la diversité au sein de l'essaim.

Dans ce cadre, le remplacement de l'opérateur de comparaison (pour déterminer si une solution *a* est meilleure qu'une solution *b*) est une modification naturelle pour adapter TRIBES à l'optimisation multiobjectif. En utilisant la dominance de Pareto pour comparer les particules, un ensemble de solutions non-dominées sera produit au niveau de chaque itération. Ces dernières seront stockées dans une archive externe, comme cela est déjà fait dans de nombreuses autres méthodes [Sierra et al., 06]. Ainsi, à la fin du traitement, l'archive

contiendra toutes les solutions non-dominées rencontrées au cours de l'exécution et, de ce fait, l'approximation du front de Pareto du problème.

Les paragraphes suivants détaillent les modifications qui ont dû être apportées à TRIBES pour le transformer en TRIBES-Multiobjectif.

3.3.1 Techniques d'archivage

Une archive externe est utilisée pour stocker les solutions non-dominées trouvées pendant le processus de recherche. Pour éviter une explosion de la complexité de l'algorithme, il est nécessaire de borner la taille de l'archive. Borner la taille de l'archive permet de gagner en complexité. En effet, lorsque l'archive est pleine, il est nécessaire qu'elle soit dotée d'un algorithme supplémentaire pour contrôler sa taille et pour décider si une solution est acceptée ou non.

La mise à jour de l'archive consiste à ajouter toutes les particules non-dominées à l'archive et à supprimer les particules dominées déjà présentes. Si le nombre de particules dans l'archive excède un nombre fixé au préalable, nous appliquons une fonction d'encombrement (*Crowd*) pour réduire la taille de l'archive et aussi maintenir sa diversité. En effet, la fonction *Crowd* utilise une grille pour découper l'espace des objectifs en un ensemble d'hypercubes. *Crowd* prend comme paramètre un entier *n* fixé au préalable et qui désigne le nombre de segments que nous devons avoir après le découpage suivant chaque objectif. En fait, si on désigne par K_i l'écart entre la plus petite valeur et la plus grande valeur suivant un objectif f_i dans l'archive, le pas de discrétisation est dans ce cas $P_i = K_i/n$ (voir Figure 3.5).

Le rôle de la fonction *Crowd* est de donner, pour chaque particule, le nombre de particules de l'archive qui occupent le même hypercube.

L'archive est mise à jour, suivant la procédure décrite par l'algorithme 9. tribeNb est le nombre de tribus dans l'essaim, tribe[i].*particuleNb* est le nombre de particules à l'intérieur de la $i^{ème}$ tribu, tribe[i].particle[j] est la $j^{ème}$ particule de la $i^{ème}$ tribu, tailleArchive est le nombre de solutions non-dominées stockées dans l'archive et tailleMaxArchive est la taille maximale de l'archive.

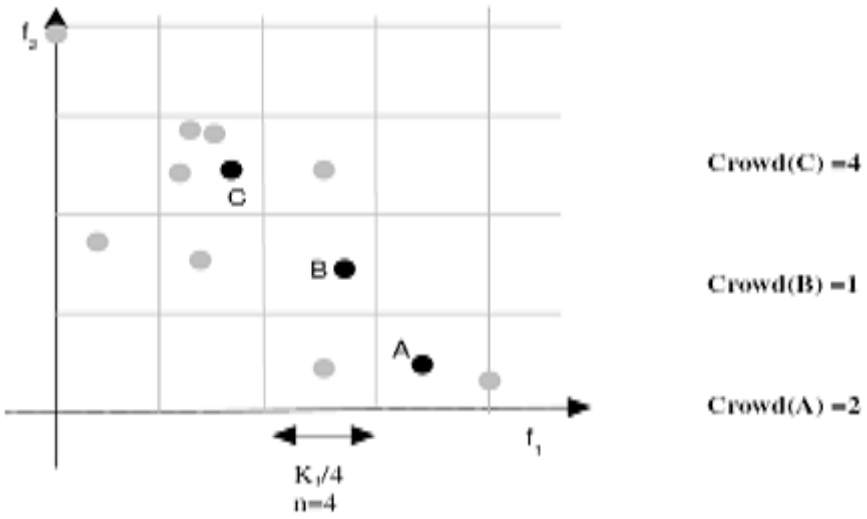


Figure 3.5: La fonction Crowd

Algorithme 9 : Mise à jour de l'archive

```
Pour i=1 à tribeNb
  Pour j=1 à tribe[i].particuleNb
    Si tribe[i].particule[j] domine des éléments de l'archive
      Effacer les éléments dominés
    Fin si
    Si tribe[i].particle[j] est non-dominé
      Si tailleArchive ≠ tailleMaxArchive
        Ajouter tribe[i].particule[j] à l'archive
      Sinon
        Calculer le Crowd de chaque solution
        Trier les éléments de l'archive en fonction de la fonction Crowd
        Remplacer la solution avec le plus grand Crowd par tribe[i].particule[j]
      Fin si
    Fin si
  Fin pour
Fin pour
```

3.3.2 Choix des informateurs

La solution à un problème d'optimisation multiobjectif est un ensemble de solutions non-dominées équivalentes. Il n'y a aucune solution qui puisse être dite meilleure que les autres. Le concept d'informatrice dans TRIBES-Multiobjectif est, de ce fait, différent de celui de TRIBES. En effet, ayant plusieurs solutions équivalentes implique l'inclusion dans l'algorithme de nouveaux critères supplémentaires pour le choix des différents guides, à

savoir le guide personnel et les informateurs. Pour gérer correctement l'inclusion d'un schéma de classification basé sur la notion de Pareto dans TRIBES, les modifications montrées ci-dessous ont été apportées :

- Pour une particule donnée, le premier informateur est *p*, l'informateur cognitif. *p* modélise la tendance de la particule à suivre sa propre expérience. A l'itération t, *p* est mis à jour si, et seulement si, la nouvelle position *x* domine *p*. Le critère de dominance est donc utilisé pour mettre à jour l'informateur cognitif.
- Le second informateur à déterminer est *g*, l'informateur social. *g* modélise l'influence de l'essaim sur le comportement de la particule. Le choix de *g* est dépendant du statut de *x* au sein de la tribu. En effet, si nous prenons une particule quelconque de la tribu, son guide est alors la meilleure particule de la tribu ou le chaman de la tribu. Si nous considérons, en revanche, le chaman d'une tribu donnée, l'informateur est alors une particule de l'archive. Ce choix étant effectué afin de donner une préférence aux solutions non-dominées. Cependant, en appliquant ces changements, la sélection de la particule informatrice appropriée devient un problème difficile depuis une archive contenant plusieurs particules équivalentes. Donc, une stratégie supplémentaire pour choisir le leader est toujours nécessaire. Quelques stratégies possibles sont les suivantes : (1) aléatoirement (un leader est aléatoirement choisi - aucune contrainte n'est imposée sur le leader), (2) Le plus proche (une particule est choisie comme leader si elle est la plus proche géographiquement de la particule considérée) et (3) un seul leader pour toutes les particules. Les différents schémas ont été testés dans [Pulido, 05] et cette étude a abouti au choix du premier schéma puisque c'est lui qui donne le meilleur compromis entre la qualité et la complexité. De plus, la sélection aléatoire de la particule informatrice peut être considérée comme un processus de diversification au sein de l'essaim.

La figure 3.4 illustre la topologie des liens d'information. Les particules noires sont les chamans des différentes tribus.

L'algorithme 10 résume la mise à jour des liens d'information dans l'essaim à l'instant t. Les éléments de cet algorithme sont les suivants :

- tribe[i].particule[j] est la $j^{ème}$ particule de la $i^{ème}$ tribu;
- tribe[i].particule[j].*p* est l'informateur cognitif de tribe[i].particule[j];
- tribe[i].particule[j].*g* est l'informateur social de tribe[i].particule[j];
- tribe[i].particule[chaman] est le chaman de la $i^{ème}$ tribu;

- Alea(archive) est une solution aléatoirement choisie dans l'archive.

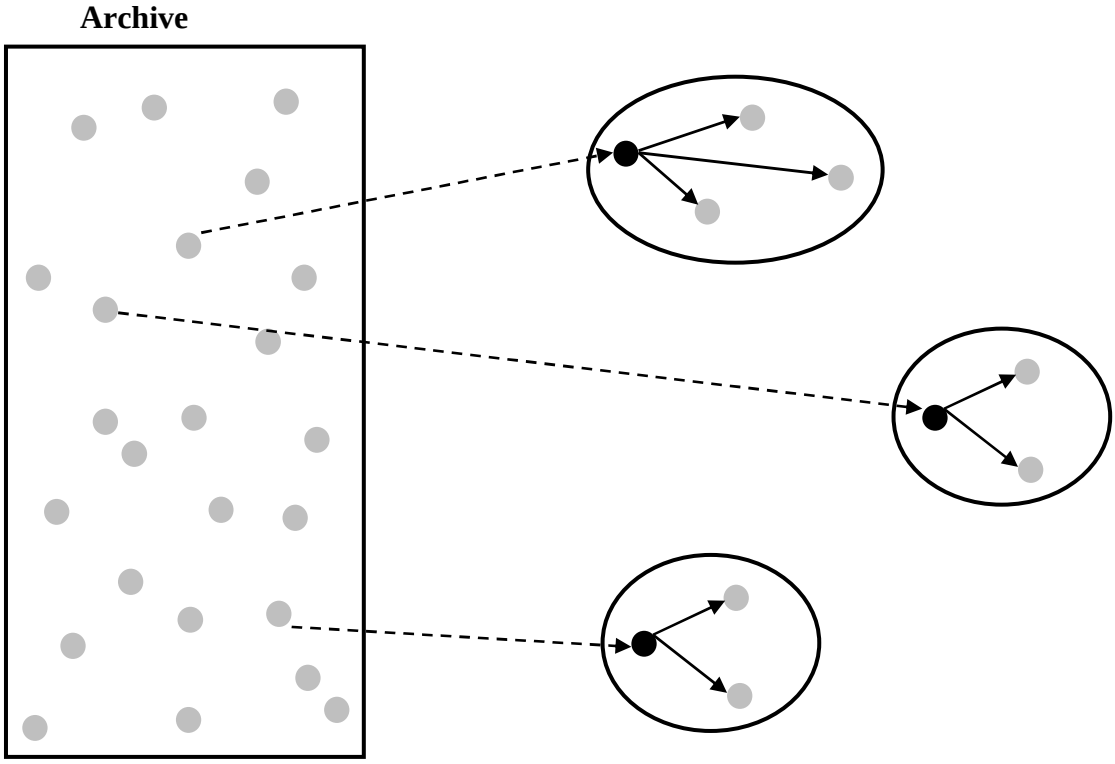


Figure 3.6: Topologie des liens d'information

Algorithme 10 : Mise à jour des liens d'information

```
Pour i=1 à tribeNb
   Pour j=1 à tribe[i].particuleNb
      Si tribe[i].particule[j] domine tribe[i].particule[j].p
         tribe[i].particule[j].p=tribe[i].particule[j]
      Fin si
      Si tribe[i].particule[j] est le chaman de la tribu i
         tribe[i].particule[j].g=Alea(archive)
      Sinon
         tribe[i].particule[j].g=tribe[i].particule[chaman].p
      Fin si
   Fin pour
Fin pour
```

3.3.3 Stratégies de déplacement

Les stratégies de déplacement sont identiques à celles de TRIBES. Elles sont résumées dans le tableau 3.2. La fonction γ est définie par l'équation (3.8).

$\gamma(x) = \frac{1}{k}\sum_{i=1}^{k} f_i(x)$, avec k le nombre des objectifs. (3.8)

Le pseudo-code de TRIBES-Multiobjectif est présenté dans la figure 3.5.

Tableau 3.2 : Stratégies de déplacement

Stratégies de déplacement	Equation
locale par gaussiennes indépendantes	$x_j = g_j + alea_{normal}(g_j - x_j, \|g_j - x_j\|), j \in \{1, \dots, D\}$
Pivot	$x = c_1.alea_{sphere}(H_p) + c_2.alea_{sphere}(H_g)$ $c_1 = \frac{\gamma(p)}{\gamma(p)+\gamma(g)}$, $c_2 = \frac{\gamma(g)}{\gamma(p)+\gamma(g)}$
pivot bruité	$x = c_1.alea_{sphere}(H_p) + c_2.alea_{sphere}(H_g)$ $b = N\left(0, \frac{\gamma(p)-\gamma(g)}{\gamma(p)+\gamma(g)}\right)$ $x = (1+b).x$

3.4 Résultats numériques de TRIBES-Multiobjectif

Après la présentation de TRIBES-Multiobjectif, nous consacrons cette partie à la comparaison de notre algorithme avec les principaux algorithmes de l'état de l'art. Pour cela, nous utilisons le jeu de fonctions de test présenté précédemment dans le premier chapitre (voir Section 1.6). Pour les fonctions à deux objectifs, la taille maximale de l'archive est fixée à 100. Pour les fonctions à trois objectifs, elle est fixée à 150. Le nombre maximal d'évaluations est fixé à 5e+4. De plus, nous avons conduit 25 exécutions indépendantes pour chaque cas.

3.4.1 Les métriques de comparaison

Dans le cas multiobjectif, les algorithmes ne peuvent plus être jugés sur la qualité de la solution fournie. De ce fait, de nombreuses métriques ont été définies, afin de juger de la qualité d'un front de Pareto. Pour notre cas, nous allons utiliser les métriques présentées avec le jeu de test de CEC'07. Cette combinaison, proposée dans [Knowles et al., 06], a pour but de mesurer la convergence et la diversité. Elle utilise deux indicateurs binaires : l'indicateur R

et l'indicateur d'hypervolume. Ces indicateurs ont été présentés précédemment dans le premier chapitre (voir section 1.7.6).

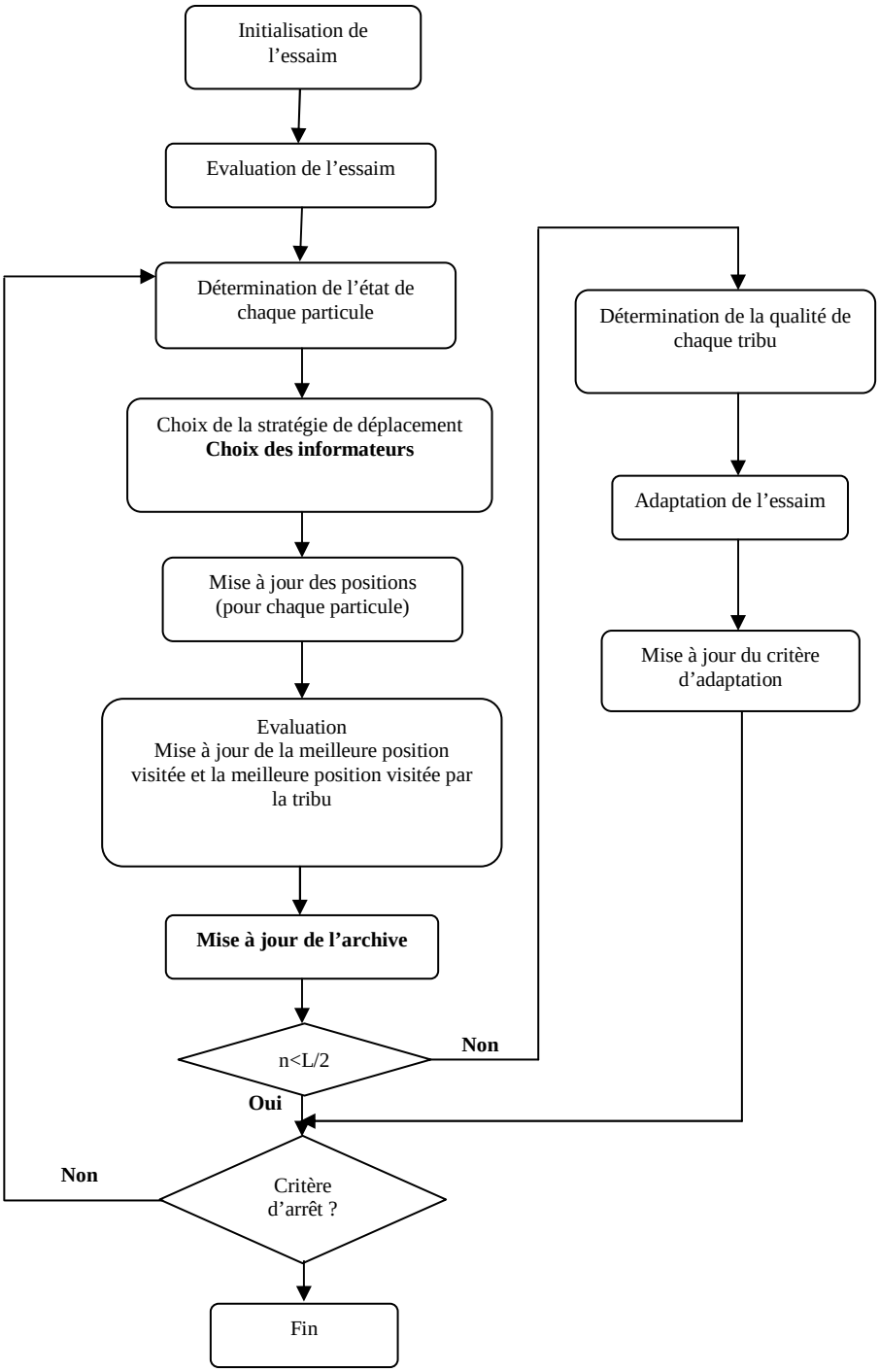


Figure 3.7: Processus de TRIBES-Multiobjectif

3.4.2 Les résultats

Les résultats obtenus avec MO-TRIBES sont comparés à ceux obtenus via NSGA-II [Deb et al., 02], qui est connu comme le meilleur algorithme d'optimisation multiobjectif, MOPSO-adaptatif et MO-TRIBES une adaptation récente de TRIBES à l'optimisation multiobjectif.

Cette adaptation se distingue de notre algorithme par l'utilisation de nouvelles règles d'adaptation pour fixer la taille de l'archive et pour réinitialiser l'essaim dans le cas d'une stagnation. De plus, la diversité de l'archive est maintenue à l'aide d'un critère basé sur la distance d'encombrement. La dernière différence réside dans les équations de déplacement utilisées.

Tableau 3.4 : L'indicateur I_R

Les fonctions	TRIBES-Multiobjectif	MO-TRIBES	MOPSO-adaptatif	NSGA-II
Oka2	8,51e-5	-1,10e-3	2,79e-2	-1,06e-3
Sympart	2,39e-4	5,18e-5	7,22e-5	1,44e-4
S_ZDT1	2,79e-3	5,12e-4	1,93e-2	1,14e-4
S_ZDT2	2,80e-4	5,01e-5	9,64e-2	4,37e-5
S_ZDT4	2,07e-3	4,96e-3	4,10e-2	8,77e-4
R_ZDT4	6,98e-3	5,23e-3	8,14e-3	2,94e-3
S_ZDT6	3,05e-3	3,51e-3	1,21e-1	2,52e-2
WFG1	**1,22e-2**	1,53e-2	7,68e-2	5,36e-2
WFG8	-4,59e-4	**-2,26e-2**	-1,30e-2	-1,73e-2
WFG9	-5,06e-3	-9,10e-3	-6,78e-3	-6,98e-3

Tableau 3.5 : L'indicateur d'hypervolume

Les fonctions	TRIBES-Multiobjectif	MO-TRIBES	MOPSO-adaptatif	NSGA-II
Oka2	-1,10e-4	-1,12e-3	5,54e-2	-1,04e-3
Sympart	1,28e-4	1,52e-4	2,09e-4	4,29e-4
S_ZDT1	2,05e-3	2,25e-3	6,27e-2	7,81e-4
S_ZDT2	2,87e-4	3,38e-4	2,25e-1	-1,15e-2
S_ZDT4	2,16e-2	2,12e-2	1,21e-1	2,39e-3
R_ZDT4	2,06e-2	1,55e-2	2,42e-2	9,26e-3
S_ZDT6	6,54e-2	**7,41e-3**	3,02e-1	5,67e-2
WFG1	3,44e-1	**8,51e-2**	3,88e-1	2,89e-1
WFG8	-2,95e-2	-1,43e-2	**-8,68e-2**	-1,04e-1
WFG9	-3,28e-2	**-5,72e-2**	-3,86e-2	-2,69e-2

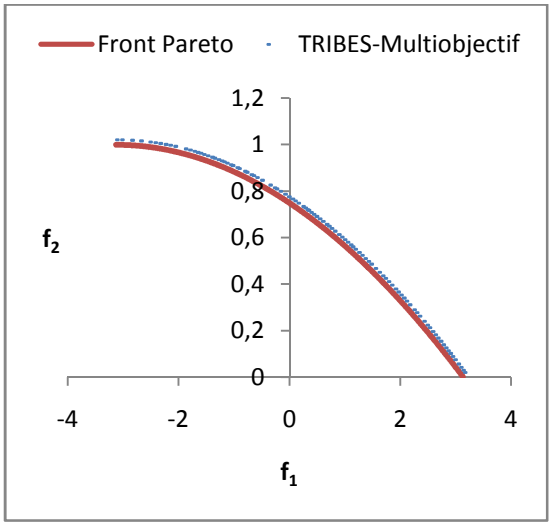


Figure 3.6 : Fronts de Pareto pour la fonction Oka2

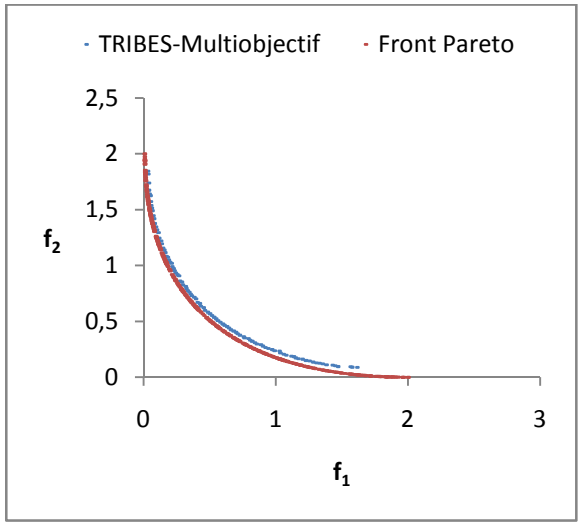


Figure 3.7 : Fronts de Pareto pour la fonction Sympart

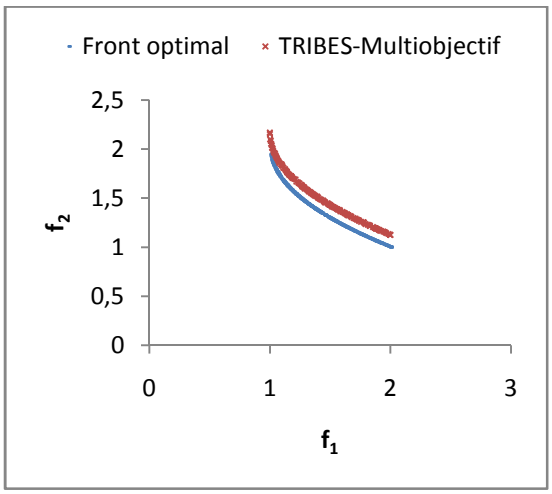


Figure 3.8 : Fronts de Pareto pour la fonction S_ZDT1

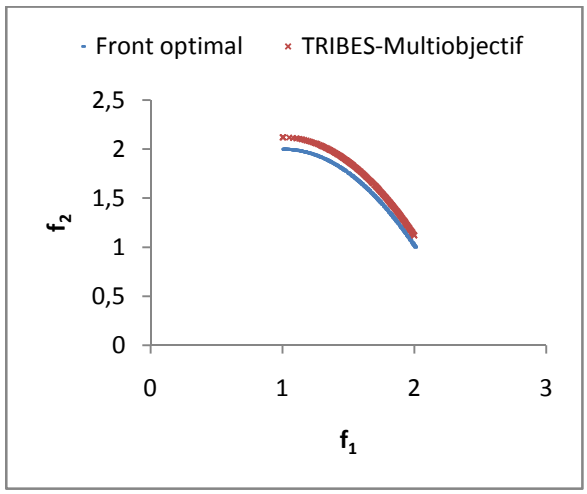


Figure 3.9 : Fronts de Pareto pour la fonction S_ZDT2

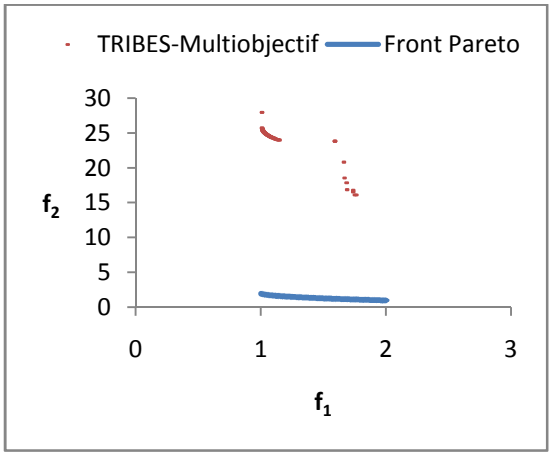


Figure 3.10 : Fronts de Pareto pour la fonction R_ZDT4

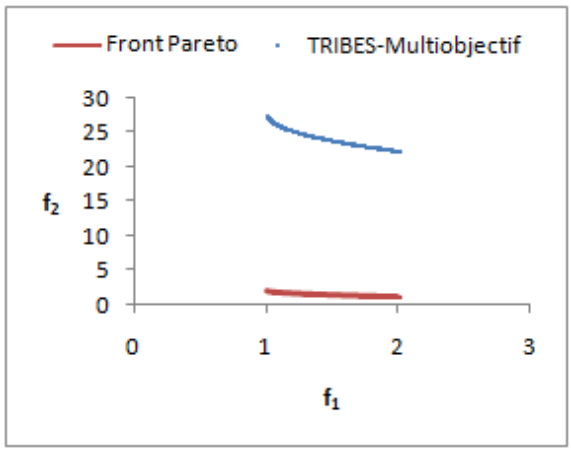


Figure 3.11: Fronts de Pareto pour la fonction S_ZDT4

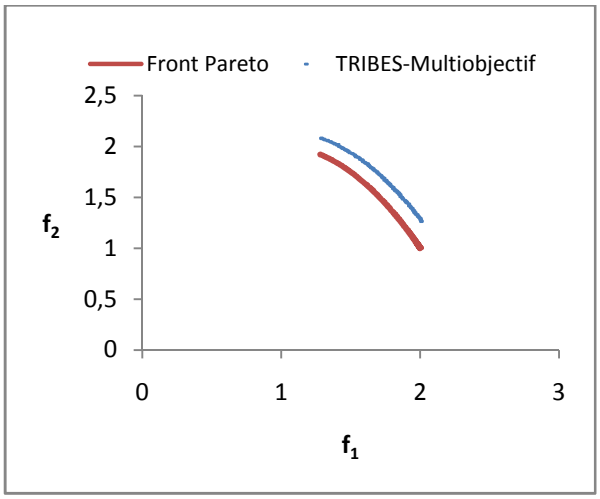


Figure 3.12 : Fronts de Pareto pour la fonction S_ZDT6

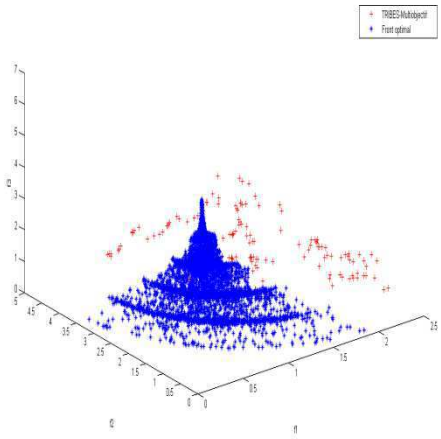


Figure 3.13 : Fronts de Pareto pour la fonction WFG1

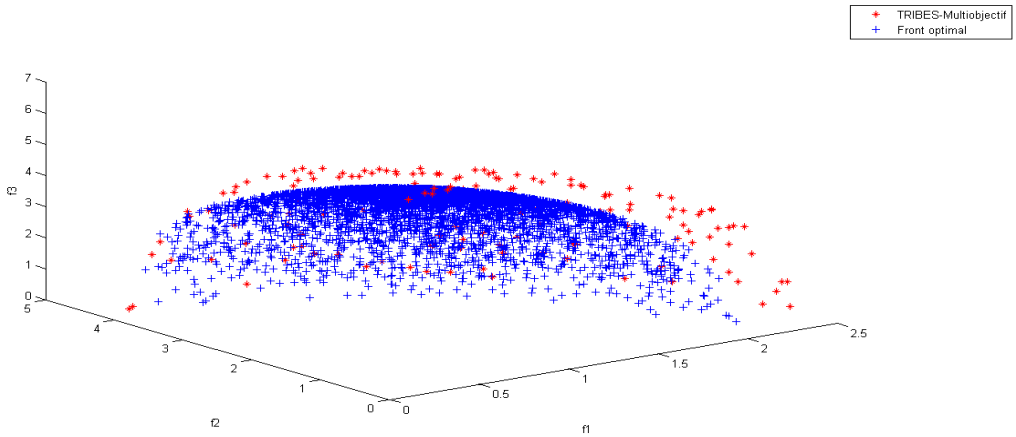


Figure 3.14 : Fronts de Pareto pour la fonction WFG8

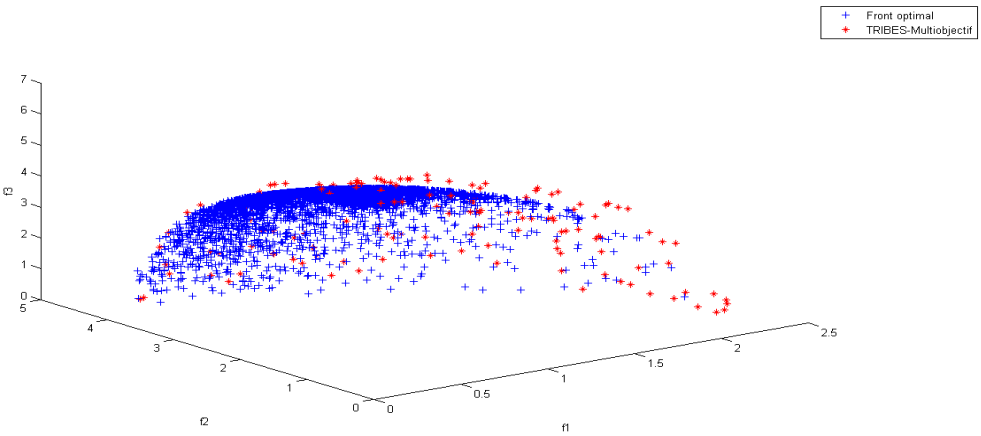


Figure 3.15 : Fronts de Pareto pour la fonction WFG9

NSGA-II est exécuté avec une population de taille 100, une probabilité de croisement de 0,8 et une probabilité de mutation de 0,1. MO-TRIBES et MOPSO-adaptatif n'ont aucun paramètre de contrôle à définir. Le critère d'arrêt utilisé pour chacun des algorithmes est le nombre maximal d'évaluations.

Les résultats concernant l'indicateur I_R sont présentés dans le tableau 3.4. Les résultats concernant l'indicateur d'hypervolume sont présentés dans le tableau 3.5. Pour chaque cas, nous présentons la moyenne des différentes 25 exécutions indépendantes.

Ce que nous constatons, essentiellement, est le fait que TRIBES-Multiobjectif et MOPSO-adaptatif donnent les plus mauvais résultats. En effet, les figures 3.6 à 3.15 montrent que TRIBES-Multiobjectif est incapable de trouver les fronts de Pareto pour la majorité des fonctions.

Les limites de MOPSO-adaptatif sont dues essentiellement au fait que cet algorithme représente une technique d'OEP classique sans mécanismes sophistiqués pour traiter le cas multiobjectif. En fait, les améliorations visent essentiellement les ajustements des paramètres de contrôle.

Les Tables 3.4 et 3.5 montrent que NSGA-II donne de bien meilleurs résultats que toutes les techniques d'OEP étudiées.

Pour étudier un peu plus les causes de ces performances de TRIBES-Multiobjectif, nous considérons les deux fonctions R_ZDT4 et S_ZDT2 présentant des fronts respectivement convexe et concave. Pour R_ZDT4, le front est également multimodal, contrairement à S_ZDT2 dont le front est monomodal. Nous avons fait varier le nombre maximal d'évaluations de la fonction objectif, à savoir 5000, 50000 et 500000. Le but est de voir l'effet de la variation du critère d'arrêt sur la qualité des fronts trouvés. Les fronts sont présentés dans la figure 3.16 et la figure 3.17.

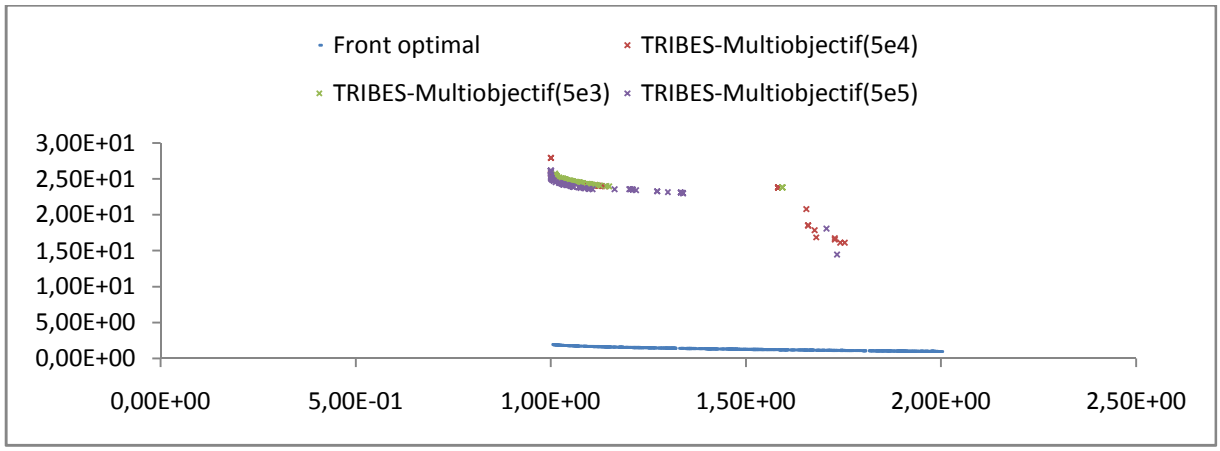


Figure 3.16 : Fronts de Pareto pour la fonction R_ZDT4 avec variation du nombre maximal d'évaluations de la fonction objectif

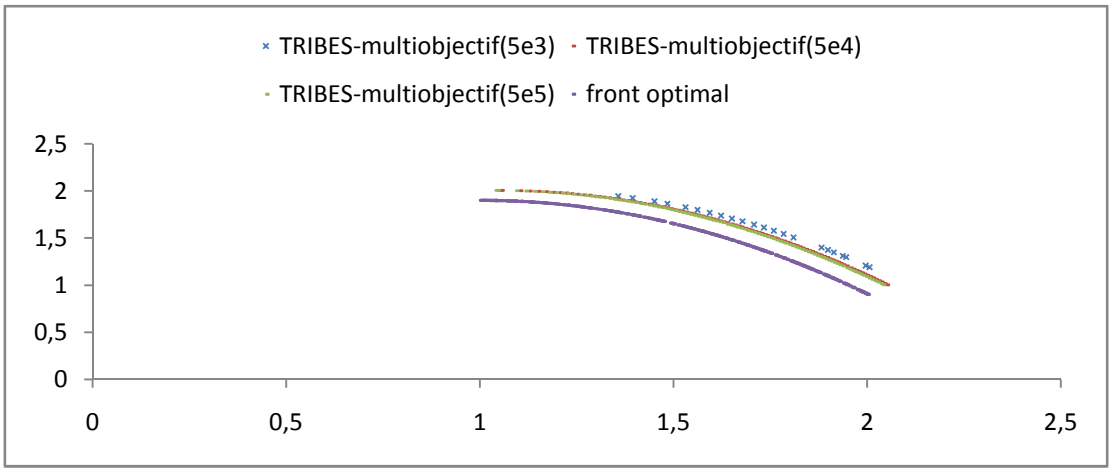


Figure 3.17: Fronts de Pareto pour la fonction S_ZDT2 avec variation du nombre maximal d'évaluations de la fonction objectif

Nous constatons une stagnation prématurée au niveau de l'essaim. Il se trouve généralement piégé au niveau des fronts locaux trouvés à partir d'un faible nombre d'itérations. Pour expliquer davantage les raisons de la très mauvaise performance de TRIBES-Multiobjectif pour la fonction R_ZDT4, nous avons suivi l'évolution de la position de la première particule créée dans l'essaim tout au long du processus de recherche. Par la suite, nous présentons l'évolution de la position et plus spécialement la quatrième variable qui varie dans l'intervalle [-5,100, 5,906] suivant le nombre d'itérations.

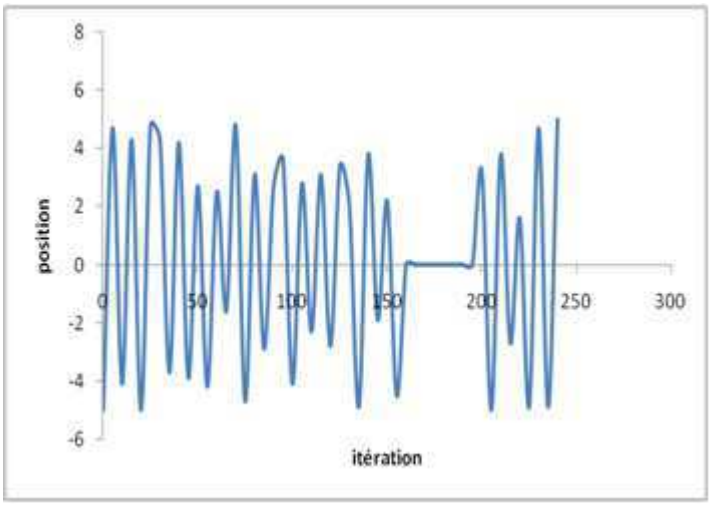


Figure 3.18 : Evolution de la position (quatrième variable) en fonction du nombre d'itérations

D'après la figure 3.18, nous constatons que cette particule subit un mouvement irrégulier alternant des valeurs très élevées avec des valeurs très basses. Une conséquence immédiate est que cette particule se déplace continuellement entre ses valeurs extrémales. De ce fait, elle ne va pas contribuer efficacement à l'exploration de l'espace de recherche.

3.5 Conclusion

Dans ce chapitre, nous avons présenté TRIBES-Multiobjectif, un algorithme adaptatif d'optimisation par essaim particulaire sans paramètres de contrôle. En effet, l'utilisation d'algorithmes adaptatifs permet aux utilisateurs de métaheuristiques de s'affranchir du problème de la définition des paramètres de l'algorithme, ceux-ci étant automatiquement calculés au cours de l'exécution de l'algorithme.

Cependant, on constate que TRIBES-Multiobjectif souffre d'un manque de diversité au sein de son essaim. En effet, il apparaît dans de nombreux cas que les particules de l'essaim se piègent très rapidement au niveau des fronts locaux. Cela entraîne une dégradation au niveau des performances de notre algorithme. Dans le chapitre suivant, nous allons présenter de nouveaux mécanismes afin de remédier à ce problème, à savoir l'hybridation de TRIBES avec des techniques de recherche locale.

Chapitre 4 Etude de l’hybridation de TRIBES-Multiobjectif avec des techniques de recherche locale

4.1 Introduction

Dans le chapitre précédent, nous avons présenté une première adaptation de TRIBES à l'optimisation multiobjectif. Cependant, TRIBES-Multiobjectif ne peut être intéressant que si la facilité de prise en main ne se fait pas au détriment de la qualité des performances.

Cependant, on constate que TRIBES souffre d'un manque de diversité au sein de son essaim. En effet, il apparaît dans de nombreux cas que la convergence de TRIBES-Multiobjectif est très rapide vers des fronts locaux et que les particules n'arrivent pas à s'échapper de ceux-ci. Cela entraîne une stagnation des particules et, donc, des performances mitigées.

Pour essayer d'apporter plus de diversité à l'essaim, nous avons introduit de nouveaux mécanismes à TRIBES-Multiobjectif, à savoir son hybridation avec une technique de recherche locale. Cette hybridation vise à augmenter l'exploitation au niveau de TRIBES.

4.2 Présentation des stratégies d'hybridation

4.2.1 Définition de la notion d'hybridation

Dans la littérature, il n'y a pas un moyen permettant de garantir qu'une méthode d'optimisation donnera les meilleurs résultats sur toutes les instances possibles, pour un problème donné. Dans la pratique, on peut toujours mettre en défaut une méthode par rapport à d'autres. L'hybridation peut être alors considérée comme un moyen efficace permettant de trouver un compromis entre les avantages et les inconvénients de plusieurs méthodes d'optimisation. Il y a plusieurs approches pour classifier les méthodes hybrides. Nous considérons par la suite celle qui considère le niveau et le mode d'hybridation.

4.2.2 Niveaux d'hybridation

Il existe deux niveaux d'hybridation : haut et bas niveaux. Le premier niveau concerne le cas où les méthodes communiquent entre elles et coopèrent ensemble mais chacune d'entre elles garde sa propre intégrité. Le deuxième niveau concerne le cas où une méthode devient un bloc fonctionnel dans une autre méthode qui l'englobe. Les figures 4.1 et 4.2 illustrent la différence entre les deux modes.

4.2.3 Modes d'hybridation

Dans la littérature, il existe deux modes d'hybridation : le mode relais et le mode co-évolutionnaire. Dans le mode relais, les méthodes d'optimisation sont exécutées d'une manière séquentielle et les résultats d'une méthode vont servir comme entrée pour l'autre. En revanche, les méthodes travaillent en parallèle dans le mode co-évolutionnaire. Les figures 4.3 et 4.4 présentent les deux différents modes.

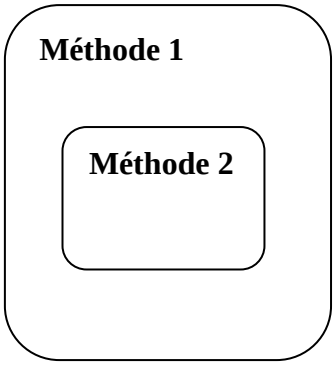


Figure 4.1 : Hybridation de bas niveau

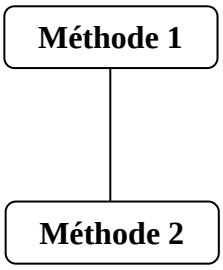


Figure 4.2 : Hybridation de haut niveau

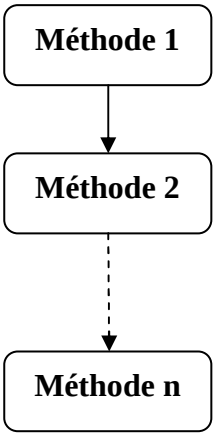


Figure 4.3 : Hybridation en mode relais

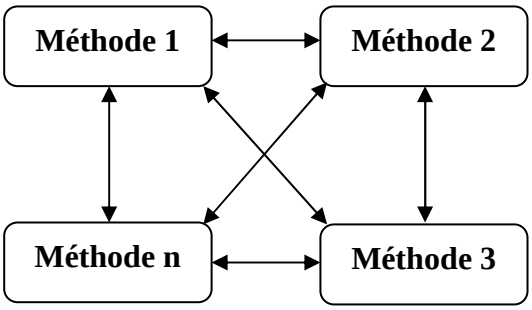


Figure 4.4 : Hybridation en mode co-évolution

Selon le niveau et le mode d'hybridation, on aura quatre classes d'hybridation possibles : relais de bas niveau, relais de haut niveau, co-évolution de bas niveau, co-évolution de haut niveau. Notons que dans la littérature, c'est la classe co-évolution de bas niveau qui a été la plus exploitée. De plus, dans cette classe, la méthode initiale garde généralement son intégrité. En revanche, pour le reste des classes, les méthodes hybridées perdent généralement leur intégrité. Dans la suite, nous proposons trois schémas différents d'hybridation de TRIBES-Multiobjectif avec des techniques de recherche locale. Ces schémas appartiennent tous à la classe co-évolution de bas niveau. En effet, les schémas proposés visent à améliorer TRIBES-

Multiobjectif sans perdre son intégrité. Les approches proposées réalisent une collaboration entre TRIBES et une méthode de recherche locale afin de profiter de leurs comportements respectifs.

4.3 Présentation

Comme tout algorithme d'optimisation par essaim particulaire, TRIBES ne peut être considéré ni comme un algorithme d'optimisation globale ni comme un algorithme d'optimisation locale. En effet, les résultats trouvés précédemment montrent que TRIBES-Multiobjectif converge rapidement vers des fronts locaux. L'essaim stagne et les particules se trouvent piégées au niveau des optima locaux. C'est pour cette raison que l'introduction d'une technique de recherche locale peut constituer une solution pour franchir les pièges des optima locaux. De ce fait, l'hybridation entre TRIBES et un algorithme de recherche locale peut être considérée comme une approche compétitive pour traiter des problèmes difficiles d'optimisation multiobjectif : pour améliorer la capacité d'exploitation de TRIBES, nous appliquons une méthode de recherche locale (R.L). L'idée générale de nos schémas d'hybridation est d'utiliser TRIBES comme algorithme principal et de lancer régulièrement une méthode de recherche locale afin d'alterner entre phases d'exploitation et phases d'exploration. Nous proposons également deux manières de lancement de la recherche locale : une manière périodique dans laquelle la recherche locale est lancée à chaque itération de TRIBES, et une manière adaptative, dans laquelle la recherche locale ne serait lancée que si une condition dépendant de l'état d'avancement de la recherche dans TRIBES était vérifiée. De plus, la recherche locale ne va pas être nécessairement appliquée d'une manière canonique, c'est-à-dire sur toutes les particules de l'essaim : nous pouvons envisager deux autres manières, à savoir l'appliquer uniquement sur la meilleure particule de chaque tribu ou bien aussi sur toutes les particules de l'archive. Les figures 4.5, 4.6 et 4.7 montrent les différentes possibilités pour appliquer la recherche locale.

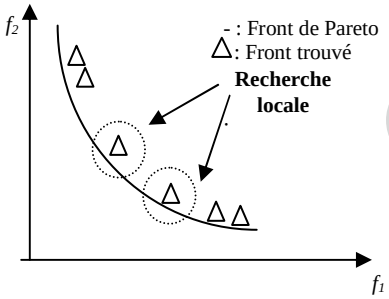


Figure 4.5 : LS-TRIBESV1

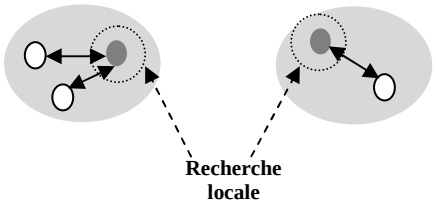


Figure 4.6 : LS-TRIBESV2

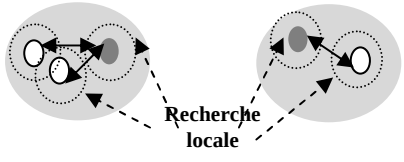


Figure 4.7 : LS-TRIBESV3

Nous aurons alors trois versions de l'algorithme. La première consiste à appliquer la recherche locale uniquement sur les particules de l'archive (LS-TRIBESV1). Cette recherche est appliquée sur les particules de l'archive situées dans les zones les moins encombrées. La deuxième version de l'algorithme consiste à appliquer la recherche locale uniquement sur les meilleures particules des tribus (LS-TRIBESV2). La troisième version consiste à appliquer la recherche locale sur tout l'essaim. Elle est effectuée au moment de son adaptation (LS-TRIBESV3).

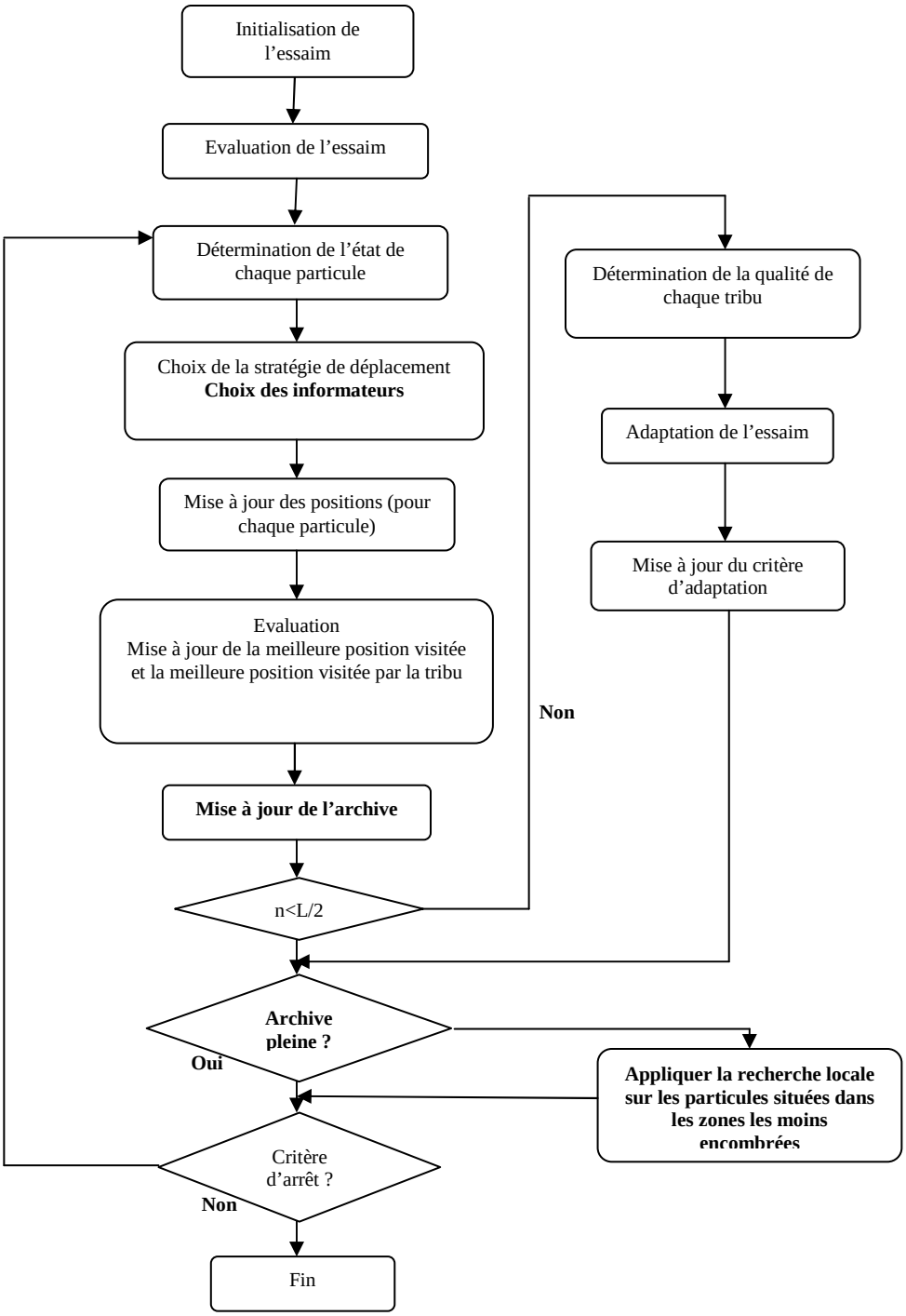


Figure 4.8 : Processus de LS-TRIBESV1

En effet, la première version consiste à appliquer la recherche locale sur les particules situées dans les zones les moins encombrées. Cette recherche n'est effectuée que lorsque l'archive devient pleine : le but est de laisser suffisamment du temps à l'information pour se propager dans l'essaim.

La deuxième version consiste à appliquer la recherche locale uniquement sur les meilleures particules des tribus. En effet, elles sont considérées prometteuses et nécessitent un travail d'intensification supplémentaire.

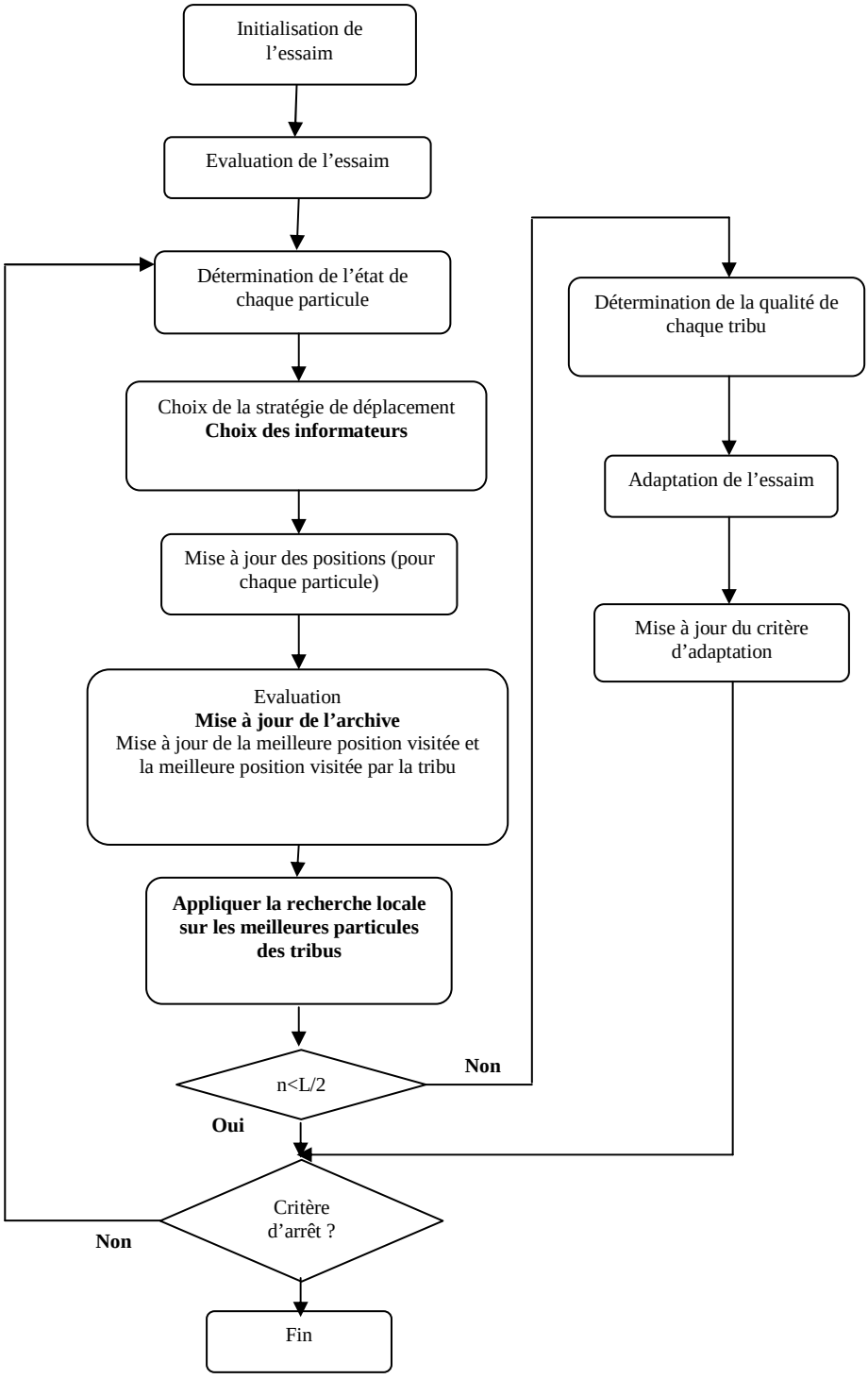


Figure 4.9 : Processus de LS-TRIBESV2

La troisième et dernière version consiste à appliquer la recherche locale d'une manière canonique sur toutes les particules de l'essaim. Cette recherche est appliquée au moment de l'adaptation de l'essaim et ceci pour laisser suffisamment de temps à l'information pour se propager dans l'essaim.

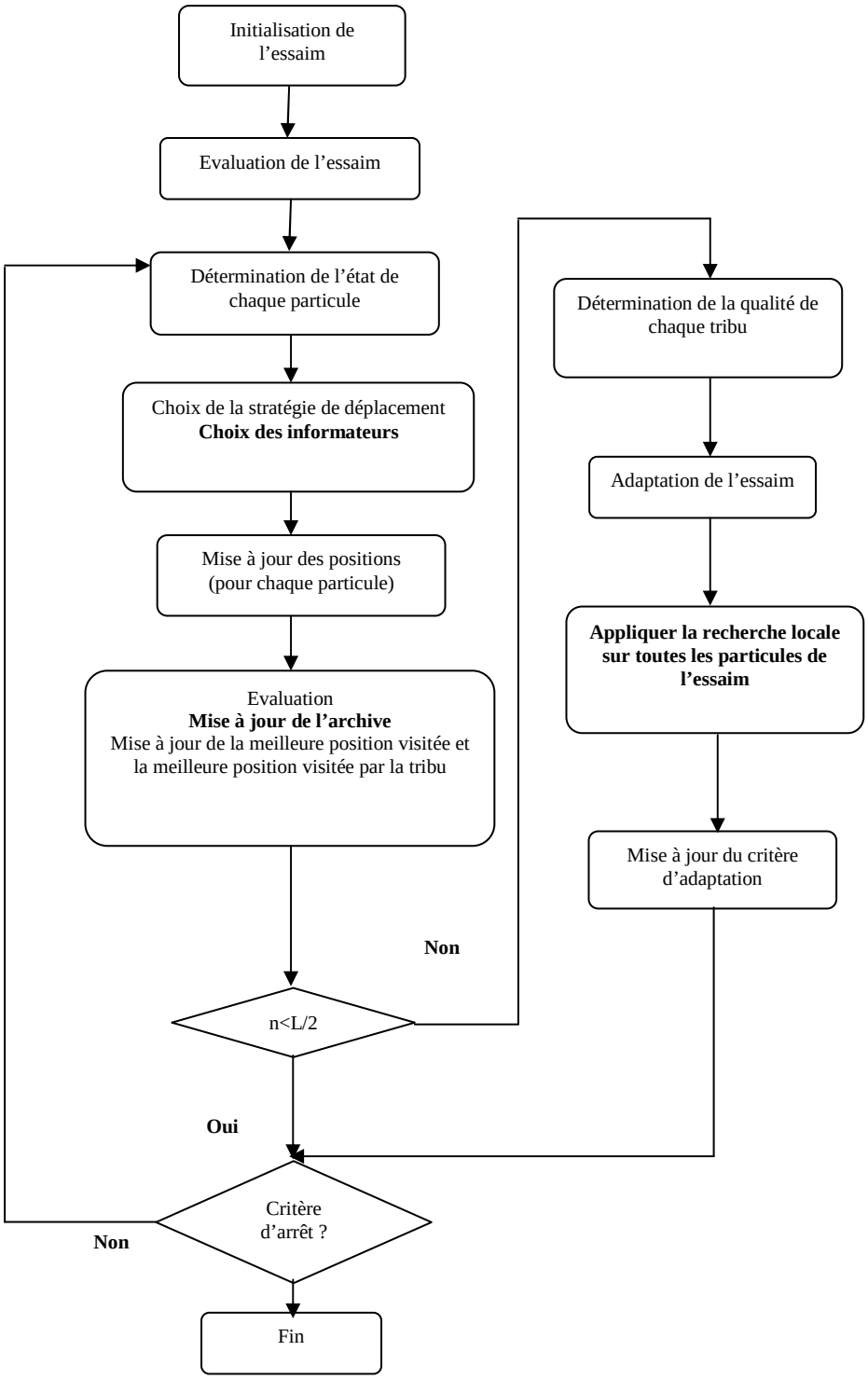


Figure 4.10: Processus de LS-TRIBESV3

4.4 Hybridation de TRIBES-Multiobjectif avec des techniques de recherche locale

Nous avons choisi d'hybrider TRIBES-multiobjectif avec deux techniques de recherche locale : le recuit simulé et la recherche tabou. En effet, ces deux métaheuristiques, contrairement à la majorité des autres techniques de recherche locale, se caractérisent par des mécanismes d'exploitation non aveugles. Elles mettent en œuvre des stratégies supplémentaires pour éviter les optima locaux. De plus, ces techniques sont simples à mettre en œuvre et ne sont pas coûteux en temps et en espace. Le but est de généraliser les résultats trouvés concernant les schémas d'hybridation proposés. Le problème au niveau de ces techniques est le fait qu'elles sont destinées essentiellement à la résolution des problèmes combinatoires. Peu de travaux ont considéré leur adaptation pour l'optimisation continue, parmi lesquels, on peut citer l'approche proposée dans [Chelouah et al., 00]. Ils proposent une adaptation de la recherche tabou à l'optimisation continue. L'approche résultante est similaire à la méthode de recherche tabou classique. La différence réside essentiellement dans la technique de génération du voisinage. En effet, il faut tout d'abord trouver un moyen pour discrétiser l'espace de recherche. Nous avons utilisé la même méthode de génération du voisinage pour le cas du recuit simulé. Nous présentons par la suite la méthode de génération du voisinage suivie pour la recherche tabou et le recuit simulé utilisés dans notre approche.

4.4.1 Le voisinage

Le voisinage est défini en utilisant le principe de boule centrée en x (solution courante) et de rayon r (elle contient alors tous les points x' avec $\|x-x'\| \leq r$ (norme euclidienne)). Pour obtenir une exploration homogène de l'espace de recherche, on considère un ensemble de boules centrées en la solution courante x avec des rayons r_0, r_1, r_2,…, r_n. L'espace est partitionné en un ensemble de couronnes concentriques $C_i(x, r_{i-1}, r_i)[=x'/ r_{i-1} \leq \|x-x'\| \leq r_i]$.
Les n voisins de x sont obtenus à l'aide d'une sélection aléatoire d'un point, dans chaque couronne [Siarry et al., 97] (voir Figure 4.11).

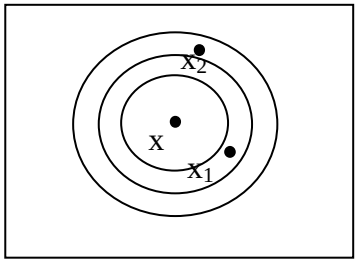


Figure 4.11 : Génération du voisinage

Les rayons sont déterminés suivant la règle suivante :

$$r_{n-i+1} = \frac{r_n \times (n-i+1)}{n}$$, i=1..*n*.

Comme LS-TRIBES a pour but d'être un algorithme multiobjectif sans paramètres de contrôle, nous avons défini des règles empiriques pour calculer r_n.

- Cas de LS-TRIBESV1 : $r_n = \frac{d_{x_{max},x_{min}}}{NBparticulesArch}$
- Cas de LS-TRIBESV2 : $r_n = \frac{d_{x_{max},x_{min}}}{NBtribe}$
- Cas de LS-TRIBESV3 : $r_n = \frac{d_{x_{max},x_{min}}}{NBparticulesEssaim}$

Avec :

- $d_{x_{max},x_{min}}$ est la distance séparant x_{max} à x_{min} ; x_{max} prend la valeur maximale suivant chaque variable ; x_{min} prend la valeur minimale suivant chaque variable,
- $NBparticulesArch$ est le nombre de particules de l'archive sur lesquelles nous appliquons la recherche locale,
- $NBtribe$ est le nombre de tribus dans l'essaim,
- $NBparticulesEssaim$ est le nombre de particules de l'essaim.

Pour le cas de LS-TRIBESV2 et LS-TRIBESV3, au début de l'exécution, nous favorisons l'exploration et vers la fin, nous favorisons l'exploitation. En effet, au début le nombre de tribus et le nombre de particules de l'essaim sont faibles ce qui induit un rayon important. Par la suite, au fur et à mesure que le processus de recherche avance, ces nombres vont croître et dans ce cas r_n diminuera.

4.4.2 La recherche tabou

4.4.2.1 Evaluation du voisinage et choix du meilleur voisin

L'évaluation du voisinage est basée sur la dominance de Pareto. Soit *x* la solution courante. La configuration voisine *x'* devant remplacer *x* représente la meilleure solution dans le voisinage. En effet, le meilleur voisin est celui qui n'est dominé par aucune solution du voisinage. Le meilleur voisin peut être retenu même s'il est plus mauvais que la solution courante. Ceci peut être considéré comme un processus de diversification.

4.4.2.2 Gestion de la liste tabou

Chaque fois qu'un mouvement est appliqué pour aller de x à x', x' est enregistré dans la liste Tabou. Ainsi, le mouvement inverse (correspondant au retour à la configuration de départ) est interdit pour les k prochaines itérations de l'algorithme. Pour implémenter la liste Tabou, nous utilisons un tableau T de n enregistrements. Chaque enregistrement contient la position x' et le rayon r_0 représentant le rayon de la plus petite boule centrée en x'.

Chaque fois que l'on examine une solution du voisinage, on vérifie si elle appartient aux boules enregistrées dans la liste tabou.

4.4.2.3 Le critère d'aspiration

Le mécanisme précédent est suffisant pour empêcher l'algorithme de rester bloqué dans des cycles courts. Cependant, un tel mécanisme peut interdire certaines configurations qui n'ont pas été encore visitées. Pour remédier à ce problème, un critère d'aspiration standard est introduit. Un mouvement marqué comme Tabou est quand même choisi si celui-ci conduit à l'obtention d'une configuration dont l'évaluation est meilleure que la meilleure configuration rencontrée jusqu'ici par l'algorithme.

Le pseudo-code de l'algorithme de recherche tabou est donné dans l'algorithme 11.

Algorithme 11 : Recherche tabou multiobjectif

$x = x_0$
Répéter
  Générer le voisinage $V(x)$ de x
  Choisir le meilleur voisin x' autorisé
  Mettre à jour la liste Tabou en fonction de x'
  $x=x'$
  Mettre à jour l'archive
FinRépéter

4.4.3 Le recuit simulé

Le schéma général du recuit simulé multiobjectif repose essentiellement sur le choix adéquat de la méthode de calcul de la probabilité d'acceptation d'une solution x'. Dans le cas multiobjectif, trois situations peuvent apparaître en comparant une solution x' à une solution x :

- *x'* domine *x* ;
- *x* domine *x'* ;
- *x* et *x'* sont équivalents.

Dans le premier cas, la probabilité d'acceptation est égale à 1. Dans le deuxième cas, elle est inférieure à 1. Dans le dernier cas, elle est indéfinie.

Le pseudo-code de l'algorithme est présenté dans l'algorithme 12.

Algorithme 12 : Recuit simulé multiobjectif

$x=x_0$
$T=T_0$
Répéter
  Générer le voisinage $V(x)$ de x
  Choisir $x' \in V(x)$
  Si x' domine x
    $x=x'$
  Sinon
    Si x domine x'
      $x=x'$ avec une probabilité $P(f(x), f(x'), T)$
    Sinon
      $x=x'$
    Fin Si
  Fin Si
  Mettre à jour l'archive
  $T=\alpha T$
Fin Répéter

Avec

- x_0 : la solution initiale ;
- x : la solution courante ;
- T_0 : la température initiale ;
- T : la température actuelle ;
- x' : le voisin retenu ;
- $V(x)$: le voisinage de x ;
- $P(f(x), f(x'), T)= \exp(-\Delta f(x,x')/T)$;
- $\Delta f(x,x')= \frac{\sum_{j=1}^{k}(f_j(x)-f_j(x'))}{k}$ avec j=1..k ; $f_j(x)$ le coût de x suivant le j^{ème} objectif ; $f_j(x')$ le coût de x' suivant le j^{ème} objectif ; k le nombre des objectifs ;
- α : il est arbitrairement choisi entre 0 et 1.

4.5 Les résultats expérimentaux

4.5.1 Etude du choix des paramètres pour les trois versions proposées

Pour valider nos approches et pour justifier l'utilisation des techniques de recherche locale, nous commençons par étudier l'impact de la taille du voisinage sur les performances des algorithmes proposés.

Pour ce faire, nous avons utilisé la procédure de test de CEC'07. Dans les expérimentations, nous avons fait varier la taille du voisinage : 5, 10 et 20. Nous avons fixé le nombre maximal d'évaluations à 5e+4. Pour chaque cas, nous avons effectué 40 exécutions indépendantes.

Les résultats concernant l'indicateur I_R sont présentés dans le tableau 4.1. Les résultats concernant l'indicateur de l'hypervolume sont présentés dans le tableau 4.2. Pour chaque cas, nous présentons la moyenne de 10 différentes exécutions indépendantes.

Une présentation détaillée des différents résultats trouvés a fait l'objet de notre publication [Smairi et al., 10]. La remarque la plus importante est le fait que les différentes versions gardent la même tendance vis-à-vis de la variation de la taille du voisinage. Nous pouvons alors conclure que le paramètre *taille* n'a pas un effet significatif sur les performances des algorithmes proposés. Pour le reste de notre travail, nous allons fixer la taille du voisinage à 10. En effet, cette valeur n'est pas trop petite, ce qui conduirait à une exploration non significative de l'espace de recherche. Elle n'est pas non plus trop grande, ce qui conduirait à une complexité grandissante.

Dans d'autres expérimentations, nous avons fait varier également le nombre maximal d'évaluations, à savoir 10e+3, 5e+4 et 10e+5. Une présentation détaillée des différents résultats trouvés est faite dans notre publication [Smairi et al., 10b]. La principale constatation est la mauvaise convergence, détectée pour un faible nombre maximal d'évaluations, pour toutes les versions sauf LS-TRIBESV3. En effet, nous pouvons l'expliquer par le fait que TRIBES commence par une seule particule et il nécessite un temps supplémentaire pour exploiter l'espace de recherche. LS-TRIBESV3 a une meilleure convergence grâce à la recherche locale qui est appliquée sur toutes les particules de l'essaim et ceci dès le début de l'exécution. C'est pourquoi, nous fixons, dans le reste de notre travail, le nombre maximal d'évaluations à 5e+4.

Table 4.1 : L'indicateur I_R

Fonctions de test	Taille du voisinage	TS-TRIBESV1	TS-TRIBESV2	TS-TRIBESV3
Oka2	**5**	**-1,22e-3**	-1,13e-3	-1,21e-3
	10	**-1,13e-3**	-1,09e-3	-1,08e-3
	20	**-1,14e-3**	-1,07e-3	-1,09e-3
Sympart	**5**	6,24e-5	**3,72e-5**	7,41e-5
	10	**3,43e-5**	4,59e-5	6,06e-5
	20	**3,95e-5**	5,01e-5	7,33e-5
S_ZDT1	**5**	**5,67e-4**	1,22e-3	1,01e-3
	10	**4,52e-4**	1,18e-3	8,64e-4
	20	**6,15e-4**	1,21e-3	1,18e-3
S_ZDT2	**5**	**4,08e-5**	1,61e-3	4,93e-5
	10	**3,03e-5**	1,27e-3	6,15e-5
	20	**3,78e-5**	1,46e-3	4,72e-5
S_ZDT4	**5**	2,67e-3	5,84e-3	**1,69e-5**
	10	2,64e-3	7,73e-3	**9,53e-6**
	20	2,76e-3	9,06e-3	**8,40e-6**
R_ZDT4	**5**	7,23e-3	2,68e-3	**1,07e-3**
	10	6,92e-3	2,78e-3	**1,82e-4**
	20	7,11e-3	2,79e-3	**1,37e-4**
S_ZDT6	**5**	3,04e-3	7,72e-3	**2,65e-3**
	10	**2,74e-3**	7,63e-3	2,97e-3
	20	3,22e-3	7,01e-3	**3,06e-3**

Table 4.2 : L'indicateur d'Hypervolume

Fonctions de test	Taille du voisinage	TS-TRIBESV1	TS-TRIBESV2	TS-TRIBESV3
Oka2	**5**	**-1,22e-3**	**-1,22e-3**	-1,17e-3
	10	**-1,21e-3**	-1,08e-3	-1,09e-3
	20	**-1,23e-3**	-1,14e-3	-1,13e-3
Sympart	**5**	1,82e-4	**9,93e-5**	2,07e-4
	10	1,35e-4	**1,08e-4**	1,83e-4
	20	1,74e-4	**1,07e-4**	1,68e-4
S_ZDT1	**5**	**1,01e-3**	4,78e-3	4,62e-3
	10	**1,45e-3**	3,92e-3	4,98e-3
	20	**1,53e-3**	3,76e-3	5,22e-3
S_ZDT2	**5**	**3,27e-4**	2,91e-3	4,02e-4
	10	**3,27e-4**	2,44e-3	4,45e-4
	20	**3,24e-4**	1,06e-3	5,13e-4
S_ZDT4	**5**	7,86e-3	2,45e-2	**1,14e-3**
	10	7,42e-3	2,12e-2	**1,57e-3**
	20	6,90e-3	2,74e-2	**1,53e-3**
R_ZDT4	**5**	1,76e-2	6,88e-3	**5,11e-3**
	10	1,82e-2	7,32e-3	**1,45e-3**
	20	2,49e-2	8,03e-3	**1,02e-4**
S_ZDT6	**5**	**6,60e-3**	1,68e-2	**1,17e-2**
	10	**7,42e-3**	1,93e-2	1,30e-2
	20	**7,91e-3**	2,05e-2	1,88e-2

4.5.2 Comparaison avec d'autres approches MOPSO

Dans la section précédente, nous avons décidé de fixer la taille du voisinage à 10 et le nombre maximal d'évaluations à 5e+4. Dans cette section, les performances des trois versions proposées ainsi que la version sans la recherche locale (TRIBES-Multiobjectif) seront comparées à MO-TRIBES, MOPSO adaptatif et NSGAII.

Les résultats concernant l'indicateur I_R sont présentés dans le tableau 4.3. Les résultats concernant l'indicateur I_H sont présentés dans le tableau 4.4. Pour chaque cas, nous présentons la moyenne des 40 différentes exécutions indépendantes. Les figures 4.12 à 4.24 représentent les fronts trouvés.

D'après ces tableaux, nous constatons que:

- Pour les fonctions S_ZDT1, S_ZDT2 et S_DTLZ2, les résultats trouvés sont très proches du front de référence (pour toutes les versions de l'approche par hybridation).
- Pour les fonctions OKA2, WFG8 et WFG9 les fronts trouvés par les différentes adaptations de TRIBES sont meilleurs que les fronts de référence.
- Pour toutes les fonctions, SMPSO et LS-TRIBESV1, avec ses deux variantes TS-TRIBESV1 et SA-TRIBESV1, donnent les meilleurs résultats sauf pour les fonctions S_ZDT4 et R_ZDT4 pour lesquelles LS-TRIBESV3 est le plus performant. Nous remarquons que cette mauvaise convergence pour ces deux fonctions a été aussi signalée dans [Hu et al., 03]. Les bonnes performances de la troisième version d'hybridation peuvent être expliquées par une bonne exploration de l'espace de recherche effectuée par la recherche locale sur toutes les particules de l'essaim.
- Les résultats de MO-TRIBES sont très proches de ceux de LS-TRIBESV1. Ceci peut être expliqué par le fait que MO-TRIBES utilise aussi une technique de recherche locale qui est appliquée seulement sur les particules de l'archive.

Finalement, nous récapitulons que LS-TRIBES est compétitif puisqu'il supporte l'intensification et la diversification. En fait, le choix des informateurs est fait pour accélérer la convergence de l'essaim vers les zones de l'espace de recherche où les particules de l'archive sont placées : ceci peut être considéré comme un processus d'intensification. De plus, la mise à jour de l'archive est faite en se basant sur la fonction d'encombrement maintenant ainsi sa diversité : on peut considérer cela comme un processus de diversification. De plus, la recherche tabou et le recuit simulé permettent aussi l'intensification et la diversification. En effet, l'exploration des voisinages permet d'intensifier la recherche dans des zones spécifiques de l'espace de recherche. En plus, leurs mécanismes comme la liste

tabou ou la variation de la température permettent d'éviter le risque du piégeage dans des solutions non Pareto.

Table 4.3 : L'indicateur I_R(*10^e-3)

Fonctions de test	LS-TRIBESV1		LS-TRIBESV2		LS-TRIBESV3		TRIBES-Multiobj	MO-TRIBES	SMPSO	NSGA-II
	SA-TRIBESV1	*TS-TRIBESV1*	*SA-TRIBESV2*	*TS-TRIBESV2*	*SA-TRIBESV3*	*TS-TRIBESV3*				
Oka2	**-1,214**	**-1,138**	-0,086	-1,092	-0,644	-1,083	0,085	-1,057	0,082	-1,062
Sympart	**0,009**	**0,034**	0,045	0,044	0,081	0,060	0,238	0,051	**0,023**	0,197
S_ZDT1	**0,402**	**0,452**	1,266	1,185	0,643	0,864	2,813	0,512	0,564	0,659
S_ZDT2	**0,028**	**0,030**	1,787	1,274	0,053	0,061	0,282	0,051	0,044	0,086
S_ZDT4	3,862	2,643	8,162	7,738	**0,006**	**0,009**	2,054	4,961	**0,007**	6,936
R_ZDT4	8,246	6,920	4,596	2,787	**0,116**	**0,182**	6,962	5,235	**0,145**	1,903
S_ZDT6	**1,883**	**2,741**	5,372	7,638	3,845	2,973	3,021	3,510	3,208	3,192
WFG1	34,201	26,104	67,216	45,418	31,108	42,240	**12,214**	15,336	24,350	**13,171**
WFG8	**-26,720**	**-18,824**	-11,227	-12,415	-4,237	-4,393	-0,459	-6.830	**-23,401**	-13,426
WFG9	**-11,995**	**-9,250**	-3,607	-6,181	-7,529	-3,927	-5,046	-9.102	-6,533	-6,204

Table 4.4 : L'indicateur d'hypervolume(*10^e-3)

Fonctions de test	LS-TRIBESV1		LS-TRIBESV2		LS-TRIBESV3		TRIBES-Multiob	MO-TRIBES	SMPSO	NSGA-II
	SA-TRIBESV1	*TS-TRIBESV1*	*SA-TRIBESV2*	*TS-TRIBESV2*	*SA-TRIBESV3*	*TS-TRIBESV3*				
Oka2	**-1,141**	**-1,212**	-0,998	-1,083	-1,060	-1,094	-0,116	-1,121	6,532	-1,047
Sympart	**0,128**	**0,135**	0,165	0,108	0,197	0,183	0,127	0,152	**0,091**	0,132
S_ZDT1	**1,474**	**1,543**	4,680	3,920	5,124	4,976	2,075	2,250	2,014	3,142
S_ZDT2	0,316	0,327	3,321	2,449	0,376	0,445	0,287	0,338	0,402	**0,261**
S_ZDT4	6,692	7,426	40,209	21,124	**1,284**	**1,572**	21,840	21,129	**1,038**	5,251
R_ZDT4	21,620	18,251	8,745	7,328	**1,108**	**1,452**	20,643	15,524	**0,520**	5,729
S_ZDT6	**5,181**	**7,425**	26,457	19,308	16,010	13,082	64,936	7,491	9,534	14,307
WFG1	**164,104**	**165,307**	197,006	227,128	144,312	223,762	**142,245**	344,934	282,107	**84,384**
WFG8	**-124,006**	**-121,224**	-70,350	-77,738	-70,509	-27,970	-29,412	-14,081	**-157,028**	-21,508
WFG9	**-38,666**	**-45,208**	-21,242	-29,462	-4,178	-6,790	-32,765	-57,283	**-47,328**	-7,742

4.6 Conclusion

Nous avons présenté de nouveaux algorithmes évolutionnaires qui hybrident TRIBES avec la recherche tabou et le recuit simulé. Cette hybridation vise à combiner la convergence rapide de TRIBES avec la bonne exploitation des voisinages conduite par la recherche tabou ou le recuit simulé. Dans ce cadre, nous avons étudié l'impact du positionnement de la recherche locale sur la performance de ces méthodes hybrides. Deux métriques ont été utilisées pour évaluer les différents algorithmes. Les résultats ont montré que l'hybridation est une approche très prometteuse pour le cas de l'optimisation multiobjectif.

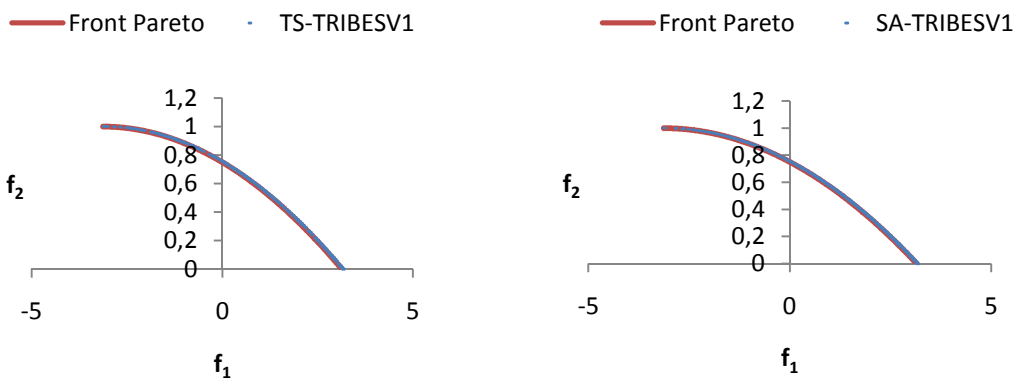


Figure 4.12 : Fronts de Pareto pour la fonction Oka2

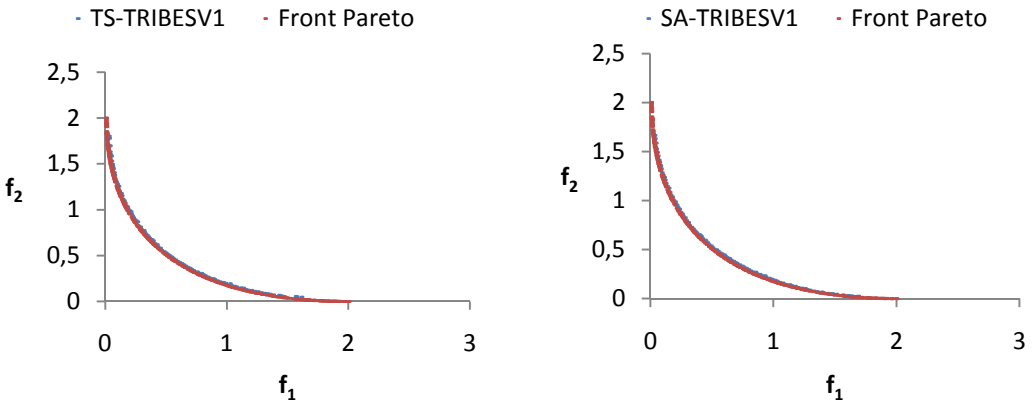


Figure 4.13 : Fronts de Pareto pour la fonction Sympart

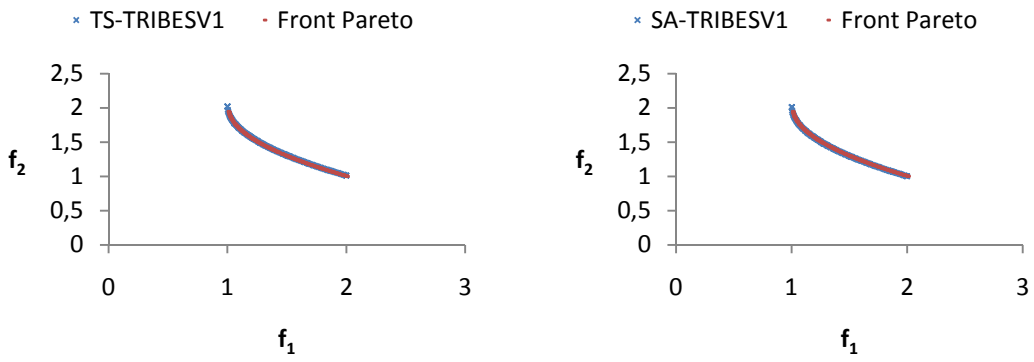


Figure 4.14 : Fronts de Pareto pour la fonction S_ZDT1

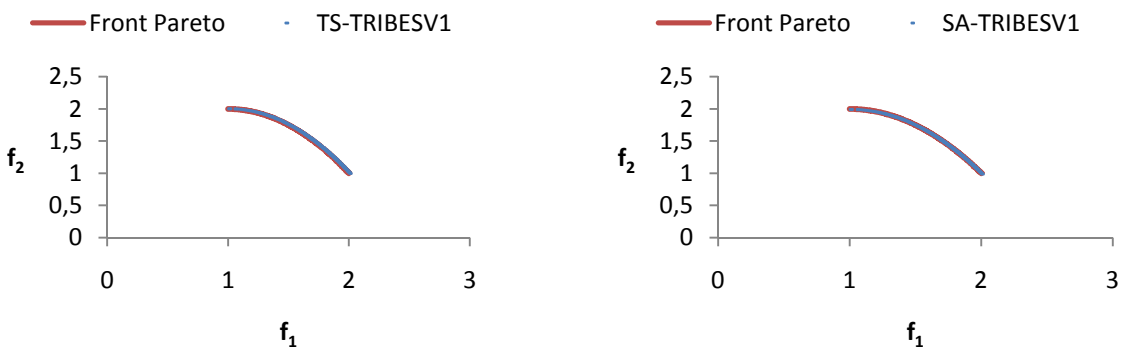


Figure 4.15 : Fronts de Pareto pour la fonction S_ZDT2

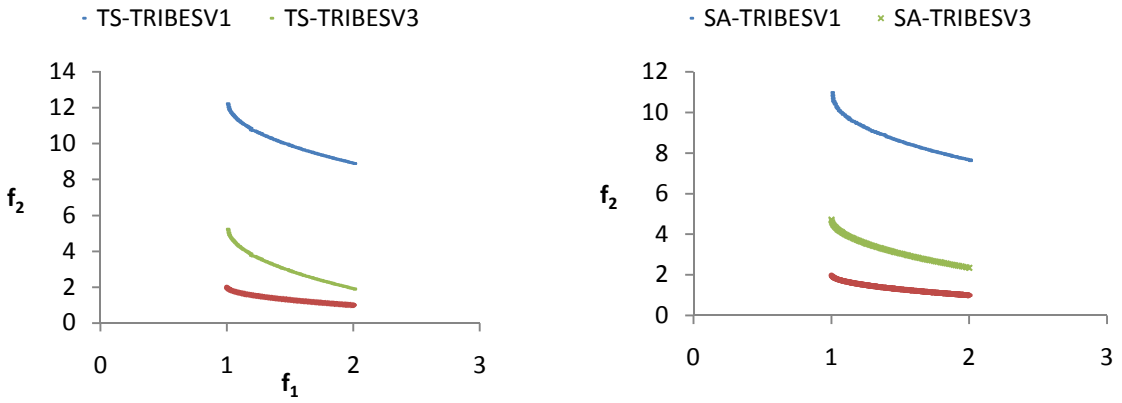


Figure 4.16 : Fronts de Pareto pour la fonction S_ZDT4

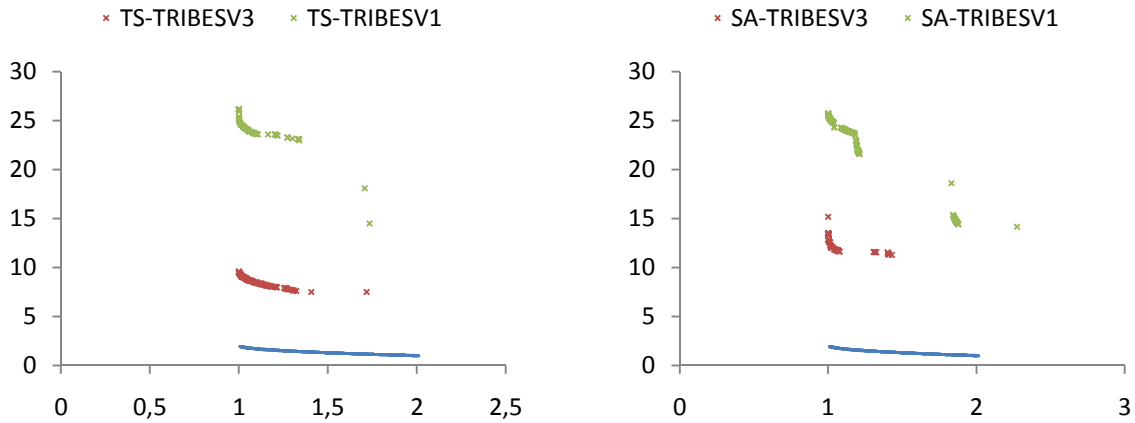


Figure 4.17 : Fronts de Pareto pour la fonction R_ZDT4

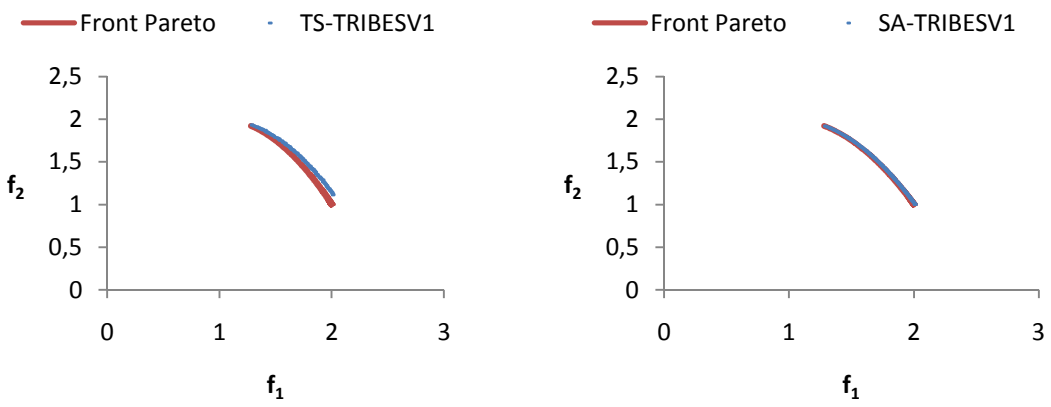


Figure 4.18 : Fronts de Pareto pour la fonction S_ZDT6

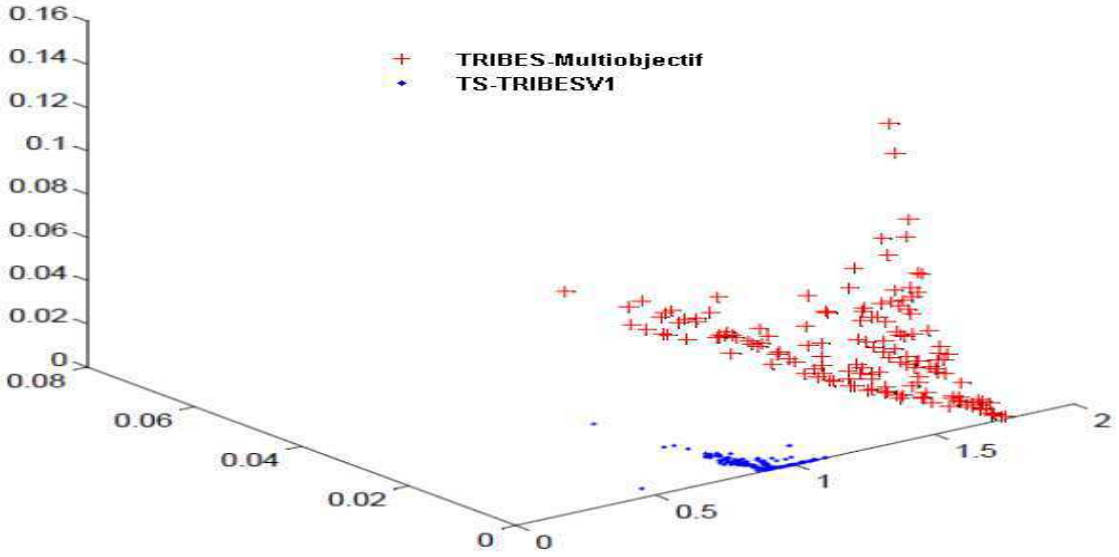


Figure 4.19 : Fronts de Pareto pour la fonction WFG1(TS-TRIBESV1)

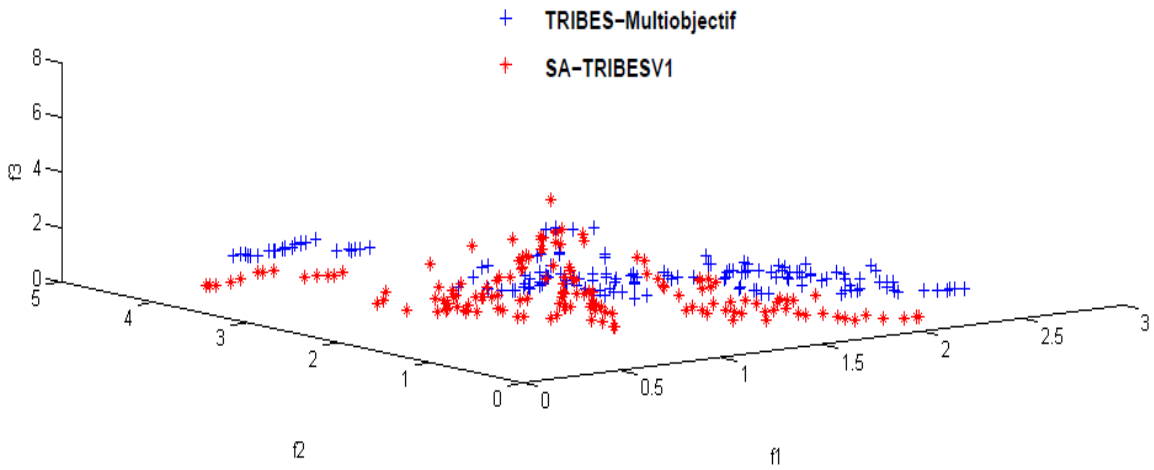


Figure 4.20 : Fronts de Pareto pour la fonction WFG1(SA-TRIBESV1)

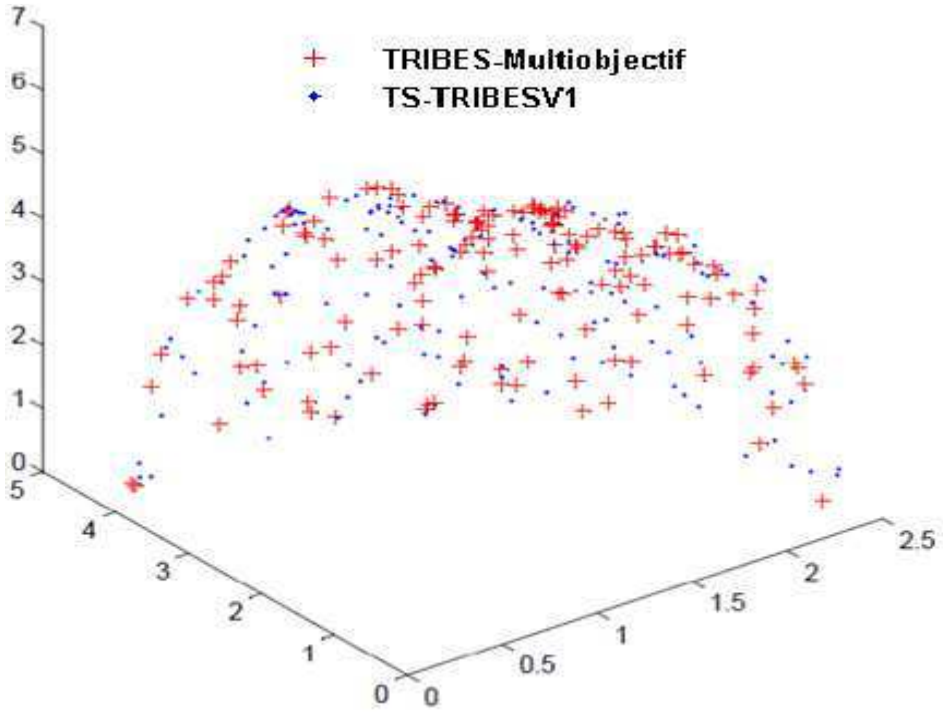


Figure 4.21 : Fronts de Pareto pour la fonction WFG8(TS-TRIBESV1)

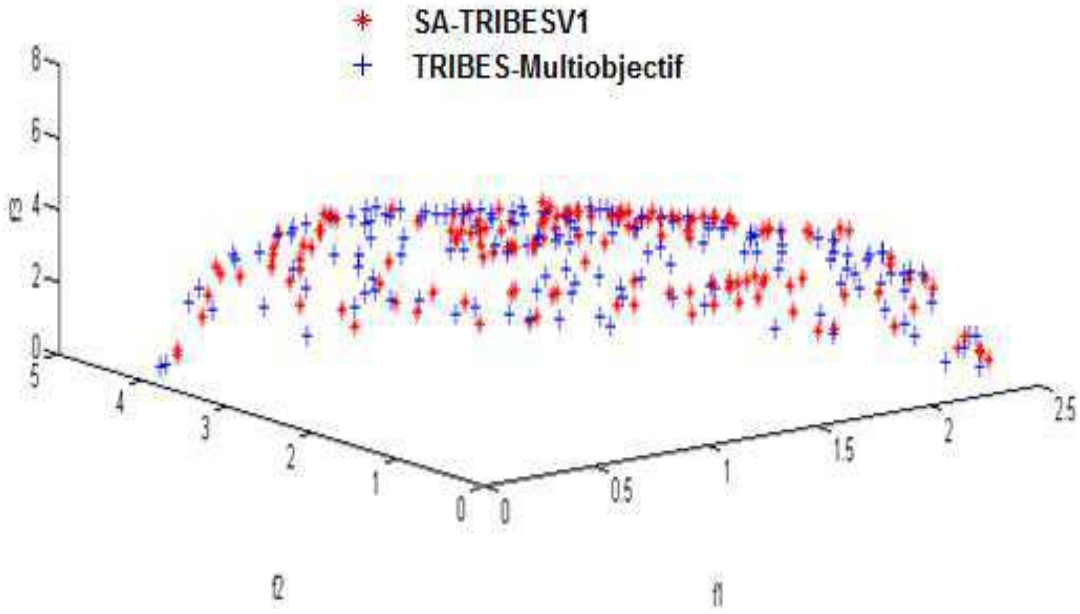


Figure 4.22 : Fronts de Pareto pour la fonction WFG8 (SA-TRIBESV1)

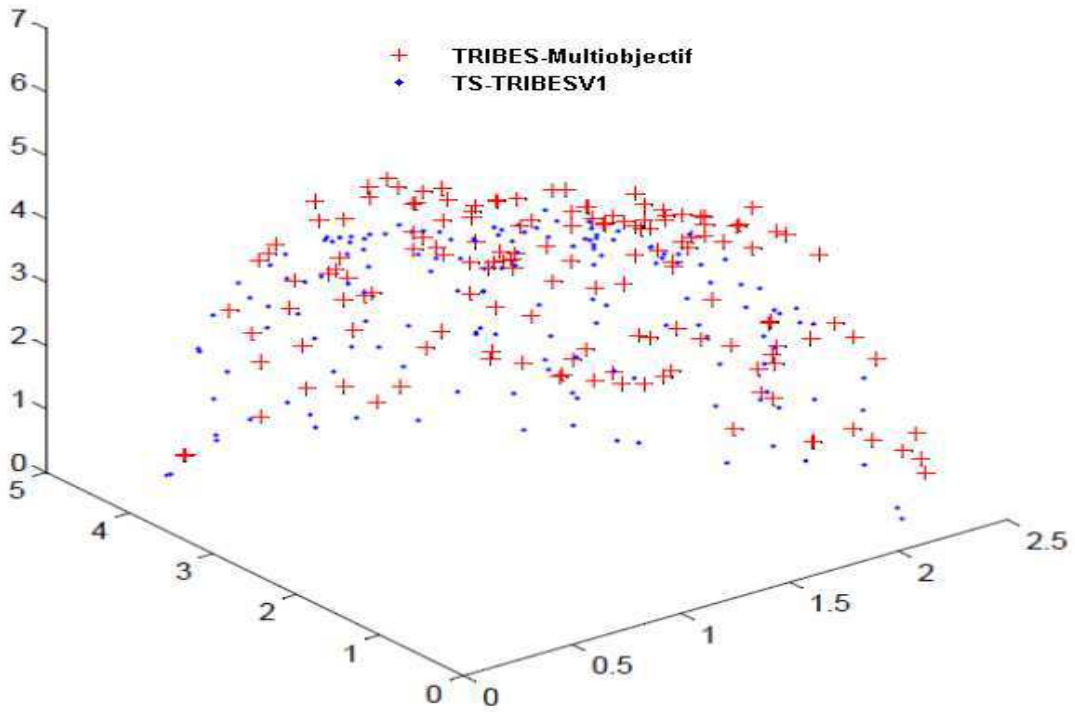


Figure 4.23 : Fronts de Pareto pour la fonction WFG9 (TS-TRIBESV1)

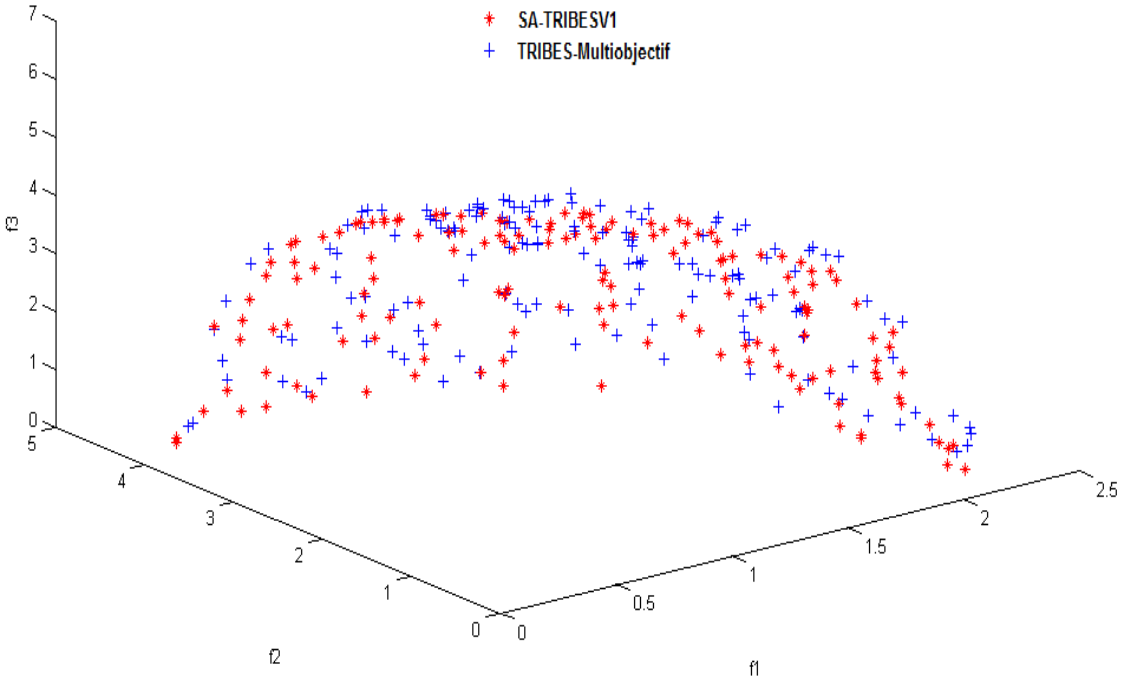


Figure 4.24 : Fronts de Pareto pour la fonction WFG9 (SA-TRIBESV1)

Chapitre 5 Application aux problèmes de dimensionnement des circuits électroniques

5.1 Introduction

La complexité des circuits intégrés VLSI suit actuellement une évolution exponentielle. Le principal moteur de cette évolution réside dans la diminution régulière de la taille de ses différents composants. D'où l'importance grandissante de l'étape de la conception des circuits intégrés qui constitue une condition nécessaire au développement d'une industrie électronique performante. Les composants analogiques tiennent une part importante dans la conception de ces circuits. En effet, le dimensionnement de ces composants est un facteur important qui agit sur les performances de ces circuits. En revanche, cette opération, généralement lente et complexe, dépend en grande partie de l'expérience et des choix du concepteur. Pour remédier à ce problème, plusieurs approches ont été proposées et qui visent à automatiser cette tâche afin de la simplifier et de l'accélérer par la suite. En effet, un problème de dimensionnement peut être perçu comme étant un problème d'optimisation à variables continues soumis à des contraintes. Les variables de décision sont alors les dimensions des transistors du circuit (c'est-à-dire longueur et largeur), les contraintes dépendent du cahier des charges du circuit. Dans ce qui suit, nous nous proposons d'appliquer LS-TRIBES aux problèmes de dimensionnement des circuits électroniques. Nous considérons deux circuits : l'amplificateur à faible bruit (LNA) et le convoyeur de courant de seconde génération. Ce travail a été effectué en collaboration avec le Laboratoire d'Electronique et des Technologies de l'Information de l'université de Sfax.

5.2 Amplificateur faible bruit

L'amplificateur à faible bruit (LNA) est un des blocs fonctionnels de base du système de communication. Le but du LNA est d'amplifier le signal reçu aux niveaux acceptables tout en réduisant au minimum le bruit qui s'ajoute. Dans cette partie, on s'intéresse à la topologie LNA à source commune avec dégénération inductive. Le choix de la structure de l'amplificateur faible bruit dépend directement des performances exigées.

L'amplificateur à source commune avec dégénération inductive présente une grande importance en termes de bruit et de fréquence. En effet, l'utilisation d'un filtre d'adaptation d'impédance à l'entrée ainsi qu'à la sortie permet d'élargir de plus la bande fréquentielle [Andreani et al., 01].

Dans ce qui suit, nous considérons la structure du LNA à source commune avec dégénération inductive, dédié pour la norme UMTS à la fréquence 2.14 GHz. Cette structure est présentée dans la figure 5.1.

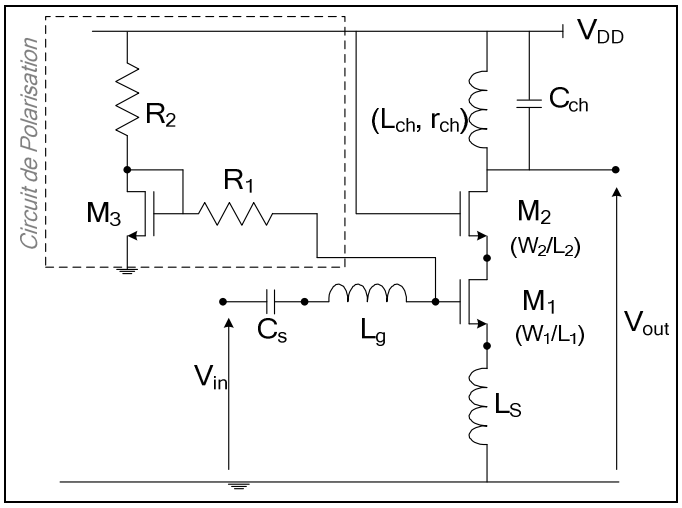


Figure 5.1: LNA source commune avec dégénération inductive dédié à la norme UMTS

La charge branchée à la sortie est un circuit LC accordé à la fréquence centrale de la norme UMTS. Le transistor M_2 garantit l'isolation entre la sortie et l'entrée de l'étage, aussi il prévient tout problème d'instabilité [Boughariou et al., 10].

La polarisation de l'étage principal est réalisée par le circuit piloté par le transistor M_3. La résistance R_2 est choisie à partir de la valeur du courant de polarisation. La résistance R_1 assure l'isolation entre le circuit de polarisation et l'amplificateur.

Le LNA à source commune avec dégénération inductive se caractérise essentiellement par l'adaptation d'impédance à l'entrée ainsi qu'à la sortie et par une valeur de gain assurant la fonction principale de l'amplification, outre le bruit et la linéarité.

Le gain de l'amplificateur faible bruit est défini comme étant le rapport entre la tension de sortie et la tension à l'entrée. Ce gain se calcule en utilisant le modèle équivalent petit signal du circuit.

La modélisation de l'adaptation d'impédance à l'entrée, de l'adaptation d'impédance à la sortie et du gain de l'amplificateur se résume dans la modélisation des paramètres S. Les expressions symboliques des paramètres sont déduites des expressions symboliques des paramètres d'impédance (Z-paramètres) du LNA, qui sont calculés en utilisant automatiquement un analyseur symbolique [Boughariou et al., 10]. En effet, le paramètre S_{11} qui est le coefficient de réflexion vu à l'entrée agit directement sur les paramètres existants à l'entrée du LNA. D'autre part, le paramètre S_{22} qui est le coefficient de réflexion vu à la sortie s'exprime en fonction des éléments constituant le réseau de sortie de l'amplificateur. Par

conséquent, les paramètres S_{11} et S_{22} conditionnent l'adaptation d'impédance à l'entrée et à la sortie respectivement. La condition d'adaptation d'impédance à l'entrée et à la sortie se résume dans les équations (5.1) et (5.2) :

$$S_{11} < -10 \text{ dB} \quad (5.1)$$

$$S_{22} < -10 \text{ dB} \quad (5.2)$$

D'un autre côté, le paramètre S_{21} représente le coefficient de transmission de la sortie vers l'entrée. Donc, la modélisation du paramètre S_{21} revient à modéliser le gain du LNA.

Le tableau 5.1 donne les relations de conversion entre S-paramètres et Z-paramètres :

Tableau 5.1 : Relations de conversion

S-paramètres	Z-paramètres
$\underline{S_{11}}$	$\frac{\left(\underline{Z_{11}} - Z_0\right)\left(\underline{Z_{22}} + Z_0\right) - \underline{Z_{12}}\ \underline{Z_{21}}}{\Delta\underline{Z}}$
$\underline{S_{21}}$	$\frac{2\underline{Z_{21}}Z_0}{\Delta\underline{Z}}$
$\underline{S_{22}}$	$\frac{\left(\underline{Z_{11}} + Z_0\right)\left(\underline{Z_{22}} - Z_0\right) - \underline{Z_{12}}\ \underline{Z_{21}}}{\Delta\underline{Z}}$

Avec $\Delta\underline{Z} = \left(\underline{Z_{11}} + Z_0\right)\left(\underline{Z_{22}} + Z_0\right) - \underline{Z_{12}}\ \underline{Z_{21}}$; $Z_0 = 50\Omega$.
Les expressions exactes des S-paramètres ne seront pas fournies vu le grand nombre de termes qu'elles contiennent.

En ce qui concerne le bruit, la figure 5.2 présente les principales sources de bruit.

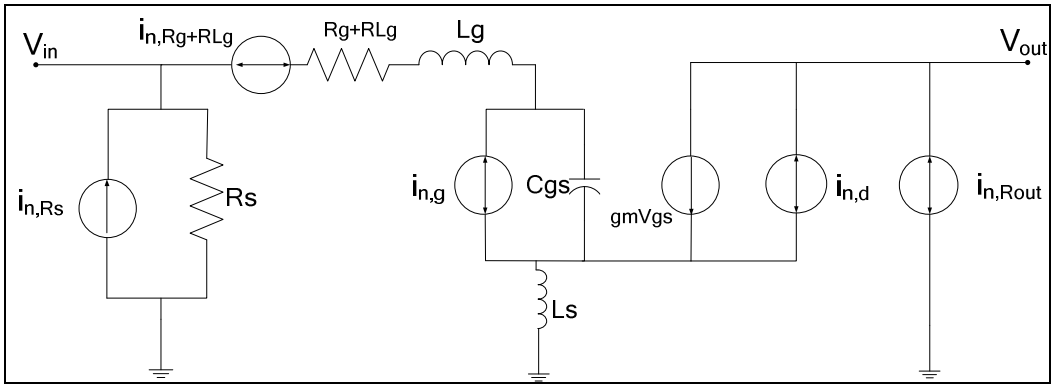


Figure 5.2: Schéma équivalent de bruit de l'amplificateur faible bruit

On note que le facteur de bruit à la fréquence de résonance est sous la forme :

$$NF = \frac{\overline{i_{n,o,R}^2} + \overline{i_{n,o,Rg+RL}^2} + \overline{i_{n,o,d}^2} + \overline{i_{n,o,g}^2} + \overline{i_{n,o,corr}^2} + \overline{i_{n,o,R_{out}}^2}}{\overline{i_{n,o,R}^2}} \quad (5.3)$$

Dans une chaîne de réception RF, le LNA doit répondre à des critères de dimensionnement stricts. Le LNA doit assurer une bonne adaptation d'impédance aussi bien à l'entrée qu'à la sortie, une valeur de gain réalisant la fonction principale d'amplification tout en assurant une valeur de figure de bruit faible. L'adaptation se traduit en termes de paramètres S par S_{11} < -10dB pour l'entrée et S_{22} < -10dB pour la sortie. Le gain se manifeste par le paramètre S_{21} et la modélisation détaillée auparavant de la figure du bruit sera considérée.

Le but de cette application est de dimensionner les transistors M_1 et M_2 de l'amplificateur faible bruit afin que celui-ci présente un gain maximal à la fréquence. Dans ce qui suit, nous présentons les résultats obtenus pour la structure de LNA proposée en appliquant les algorithmes TS-TRIBESV1 et SA-TRIBESV1. Nous avons choisi d'appliquer ces deux algorithmes puisqu'ils ont donné les meilleurs résultats (voir Chapitre 4). Une comparaison des résultats théoriques avec les résultats obtenus par le simulateur ADS qui permet de valider l'approche d'optimisation sera aussi détaillée.

Les variables du problème sont les dimensions des transistors M_1 et M_2 et le courant de polarisation avec :

- L = 0.35µm
- W ∈ [300 ,1000] µm
- I_d ∈ [10, 60] mA

L'approche symbolique [Fakhfakh et al., 10], permettant d'obtenir les expressions symboliques des paramètres Z et par la suite la déduction des paramètres S, est utilisée dans l'approche L S-TRIBESV1 pour l'optimisation des performances de cette structure.

Le tableau 5.2 présente le cahier des charges de cet amplificateur.

Tableau 5.2 : Cahier des charges du LNA à source commune dédié à la norme UMTS

S_{11} (dB)	< -10
S_{21} (dB)	> 14
S_{22} (dB)	< -10
NF (dB)	< 4

La figure suivante représente les fronts de Pareto obtenus par les deux algorithmes :

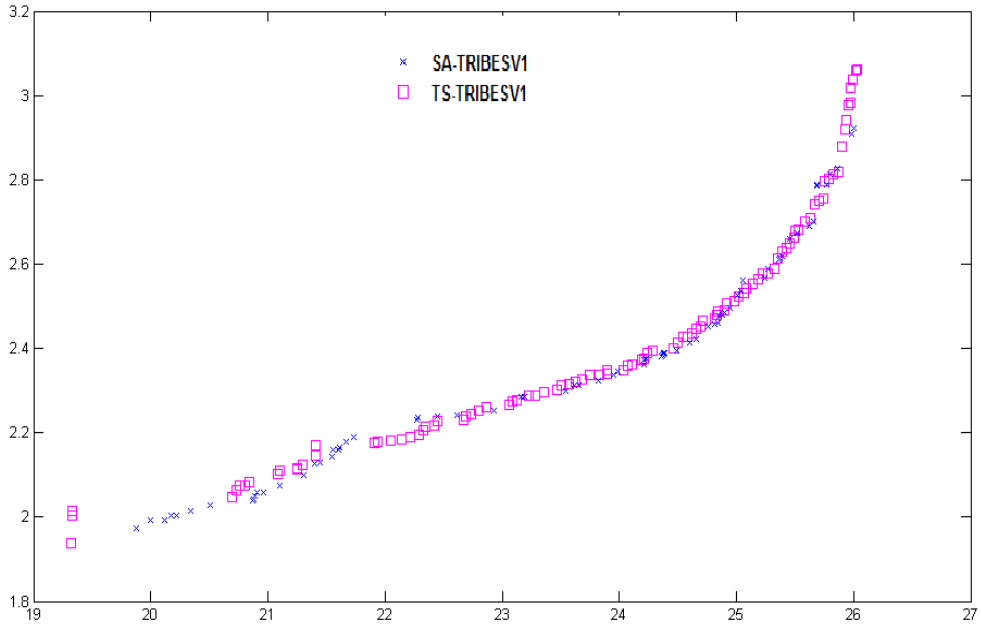


Figure 5.3 : Résultats d'optimisation du LNA à source commune dédié à la norme UMTS

Nous constatons que TS-TRIBESV1 donne le meilleur front, sachant que les critères de comparaison sont la convergence et la largeur du front. Nous présentons dans ce qui suit les résultats de simulation qui correspondent aux deux solutions optimales appartenant au front de Pareto trouvé par l'algorithme TS-TRIBESV1. En effet, ces deux solutions correspondent aux deux extrémités du front de Pareto trouvé : la première solution présente le bruit minimal et la deuxième solution présente le gain maximal à la fréquence.

Résultats de simulation de la solution 1 :

Les dimensions obtenues par la première solution sont affichées dans le tableau 5.3 :

Tableau 5.3 : Solutions optimales de la première solution du LNA à source commune

W_1(µm)	L_1(µm)	W_2(µm)	L_2(µm)	W_3(µm)	L_3(µm)	I_d(mA)	**S21(dB)**	19,32
990,18	0,35	520,83	0,35	40,00	0,35	59,80		
C_s(pF)	**L_g(nH)**	**L_s(nH)**	**L_{ch}(nH)**	**R_{ch}(Ω)**	**C_{ch}(pF)**	**R_1(kΩ)**	**NF(dB)**	1,94
10,00	3,02	0,28	0,82	1,33	7	1,00		

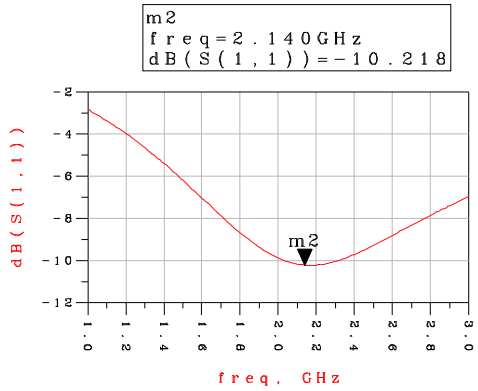


Figure 5.4: Courbe de simulation S_{11}=f(freq)

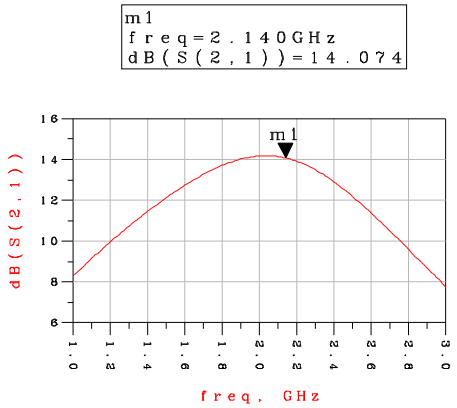


Figure 5.5: Courbe de simulation S_{21}=f(freq)

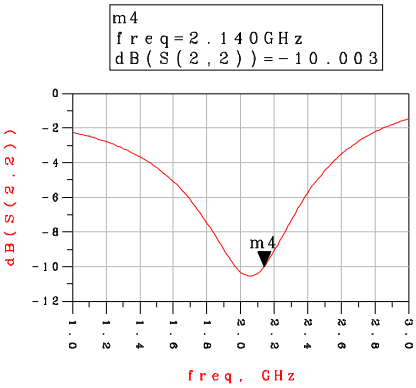


Figure 5.6: Courbe de simulation S_{11}=f(freq)

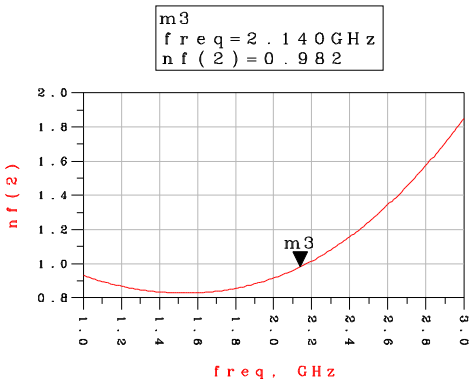


Figure 5.7: Courbe de simulation NF=f(freq)

Les résultats de simulation des paramètres S et de la figure de bruit du LNA dédié à la norme UMTS sont illustrés par les figures 5.4, 5.5, 5.6 et 5.7.

D'après la figure 5.4, nous remarquons que le LNA est bien adapté à l'entrée. La courbe présentée dans la figure 5.6 montre une valeur du gain du LNA de l'ordre de 14.07 dB.

Nous remarquons que le LNA est bien adapté à la sortie vu que S_{22} < -10dB (voir Figure 5.6). La figure de bruit est montrée dans la figure 5.7. Elle est de l'ordre de 0,98 dB qui est inférieur à 4 dB.

Résultats de simulation de la solution 2 :

Les dimensions obtenues par la deuxième solution sont affichées dans le tableau 5.4 :

Tableau 5.4 : Solutions optimales de la deuxième solution du LNA à source commune

W_1(µm)	L_1(µm)	W_2(µm)	L_2(µm)	W_3(µm)	L_3(µm)	I_d(mA)	S21(dB)	26,03
510,96	0,35	933,48	0,35	40,00	0,35	59,97		
C_s(pF)	L_g(nH)	L_s(nH)	L_{ch}(nH)	R_{ch}(Ω)	C_{ch}(pF)	R_1(kΩ)	NF(dB)	3,06
10,00	9,93	0,086	0,83	1,27	7	1,00		

Les résultats de simulation des paramètres S et de la figure du bruit du LNA dédié à la norme UMTS sont illustrés par les figures 5.8, 5.9, 5.10 et 5.11. Ce résultat montre aussi la bonne adaptation du LNA à l'entrée (voir Figure 5.8). La courbe présentée dans la figure 5.9 montre une valeur du gain du LNA de l'ordre de 17,04 dB.

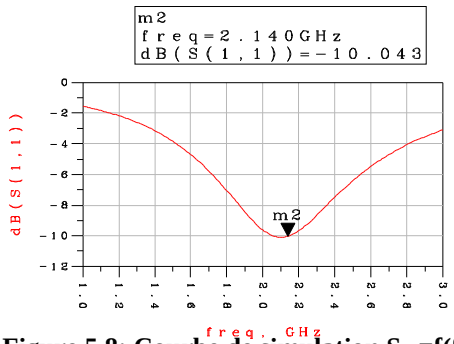


Figure 5.8: Courbe de simulation S_{11}=f(freq)

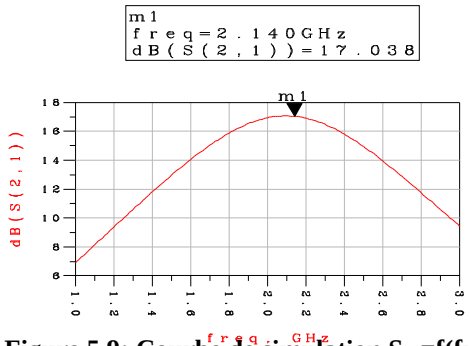


Figure 5.9: Courbe de simulation S_{21}=f(freq)

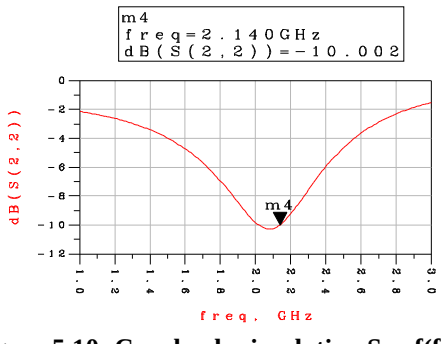


Figure 5.10: Courbe de simulation S_{22}=f(freq)

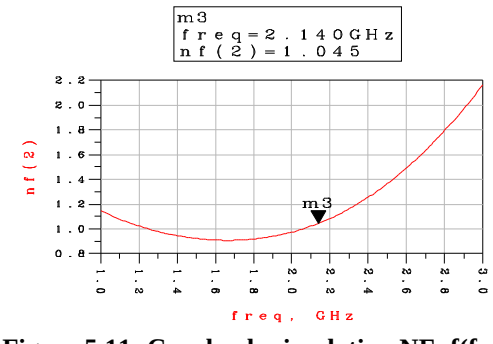


Figure 5.11: Courbe de simulation NF=f(freq)

Le LNA est bien adapté à la sortie vu que S_{22} < -10dB. D'après la figure 5.11, nous avons une figure de bruit de l'ordre de 1,05 dB qui vérifie l'exigence imposée par le cahier des charges.

Récapitulation et comparaison :

Le tableau 5.5 présente une comparaison entre les résultats théoriques et les résultats de simulation pour les 2 solutions simulées avec ADS.

Tableau 5.5 : Comparaison entre les résultats théoriques et les résultats de simulation du LNA à source commune

	Solution 1		***Solution 2***	
	Optimisation	***simulation***	***Optimisation***	***simulation***
S21(dB)	19,32	14,07	26,03	17,04
NF(dB)	1,94	0,98	3,06	1,05

En effectuant une comparaison entre les résultats obtenus par l'approche d'optimisation adoptée et les résultats de simulation, nous constatons la bonne concordance entre les résultats théoriques et les résultats de simulation de la figure de bruit. Ceci montre que l'algorithme d'optimisation TS-TRIBESV1 est performant.

La différence existante entre les valeurs de S_{21} théoriques et S_{21} simulées est due aux effets parasites caractérisant les circuits RF et qui ne sont pas pris en considération lors de la modélisation.

En outre, nous avons montré que l'approche adoptée de dimensionnement et d'optimisation permet de faciliter la tâche de la conception de l'amplificateur avec des dimensions à priori optimales.

5.3 Convoyeur de courant

Un convoyeur de courant est un circuit électronique composé de trois ports actifs. La représentation conventionnelle est donnée sur la figure 5.12.

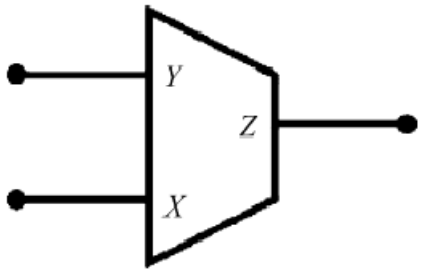


Figure 5.12: Convoyeur de courant

Les courants et tensions de ces différentes bornes sont liés par l'équation suivante :

$$\begin{pmatrix} I_Y \\ V_X \\ I_z \end{pmatrix} = \begin{pmatrix} 0 & k_2 & 0 \\ 1 & 0 & 0 \\ 0 & k_1 & 0 \end{pmatrix} \begin{pmatrix} V_Y \\ I_X \\ V_Z \end{pmatrix} \quad (5.4)$$

Dans cette équation, k_2 désigne le type de convoyeur utilisé. En effet, si k_2=0, alors le convoyeur utilisé est de première génération. Dans le cas contraire, si k_2=1, alors le convoyeur est de deuxième génération. De plus, k_1 spécifie le transfert de courant entre X et Z. Si k_1=-1, alors le transfert de courant est négatif. Dans le cas contraire, si k_1=1, le transfert est positif.
Le convoyeur de courant assure les deux fonctions suivantes : suiveur de courant entre X et Z et suiveur de tension entre X et Y.

Le convoyeur de courant de seconde génération utilise une boucle translinéaire [Seevinck, 00]. La figure 5.13 illustre l'exemple d'un convoyeur de courant de seconde génération positif à boucle translinéaire CMOS [Schmid, 00]. Les transistors M1 à M4 constituent la boucle translinéaire qui assure le suivi de tension entre X et Y. I_0 désigne le courant de polarisation et les transistors M9 à M13 représentent des miroirs de courant. Les transistors M5 à M8 servent à reproduire au port Z le courant appliqué au port X (voir Figure 5.13).

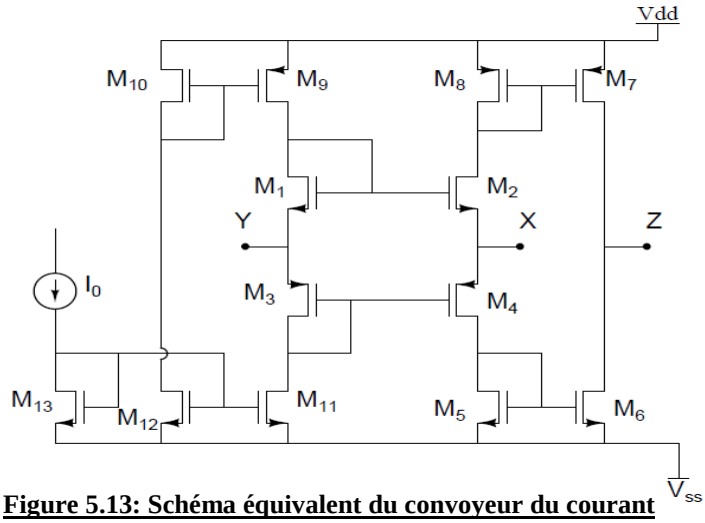


Figure 5.13: Schéma équivalent du convoyeur du courant

En réalité, le comportement du convoyeur peut être différent du cas idéal. Il y a des composants parasites qui peuvent affecter son comportement. La figure représente ces composants. En pratique, il a été prouvé que le R_x représente le composant qui a les effets les plus significatifs sur le comportement du convoyeur.

Dans les circuits à haute fréquence, les performances du convoyeur doivent rester en accord avec les choix du concepteur. Pour ce faire, le circuit doit être dimensionné précisément. Dans ce qui suit, nous nous intéressons au problème du dimensionnement du circuit tout en minimisant R_x et en maximisant la fréquence de coupure f_c.

L'expression de R_x a été élaborée dans [Seevinck, 00]. Elle est donnée par l'équation suivante :

$$R_X = \frac{1}{\sqrt{2K_N\left(\frac{W_N}{L_N}\right)(1+\gamma_N V_{DS})I_0}+\sqrt{2K_P\left(\frac{W_P}{L_P}\right)(1+\gamma_P V_{DS})I_0}} \tag{5.5}$$

où γ_N, γ_P , K_N = 0,9386.10^{-8} et K_P = 0,3476.10^{-6} sont des paramètres liés à la technologie des transistors (AMS 0,35 µm). I_0 =100 µA est le courant de polarisation et V_{DS} est la tension drain-source des transistors. W_N et L_N sont respectivement la largeur et la longueur des transistors NMOS. W_P et L_P sont respectivement la largeur et la longueur des transistors PMOS.

L'expression de la fréquence de coupure f_c ne sera pas présentée vu le grand nombre de termes qu'elle contient.

Les transistors opèrent en mode saturation. Ce qui induit un certain nombre de contraintes :

$$\frac{V_{DD}}{2} - V_{TN} - \sqrt{\frac{I_0}{K_N\frac{W_N}{L_N}}} > \sqrt{\frac{I_0}{K_P\frac{W_P}{L_P}}} \tag{5.6}$$

$$\frac{V_{DD}}{2} - V_{TP} - \sqrt{\frac{I_0}{K_P\frac{W_P}{L_P}}} > \sqrt{\frac{I_0}{K_N\frac{W_N}{L_N}}} \tag{5.7}$$

où V_{TN}= 0,4655 V et V_{TP} = 0,617 V sont des paramètres inhérents à la technologie utilisée ; V_{DD} est la source de tension de 2,5 V.

Le but est de minimiser R_x et de maximiser f_c en fonction des dimensions des transistors W_N, L_N, W_P et L_P tout en respectant les contraintes de saturation et en respectant les contraintes matérielles suivantes :

$$35.10^{-8} \leq L_N, L_P \leq 45.10^{-8} \tag{5.8}$$

$$1.10^{-6} \leq W_N, W_P \leq 30.10^{-6} \tag{5.9}$$

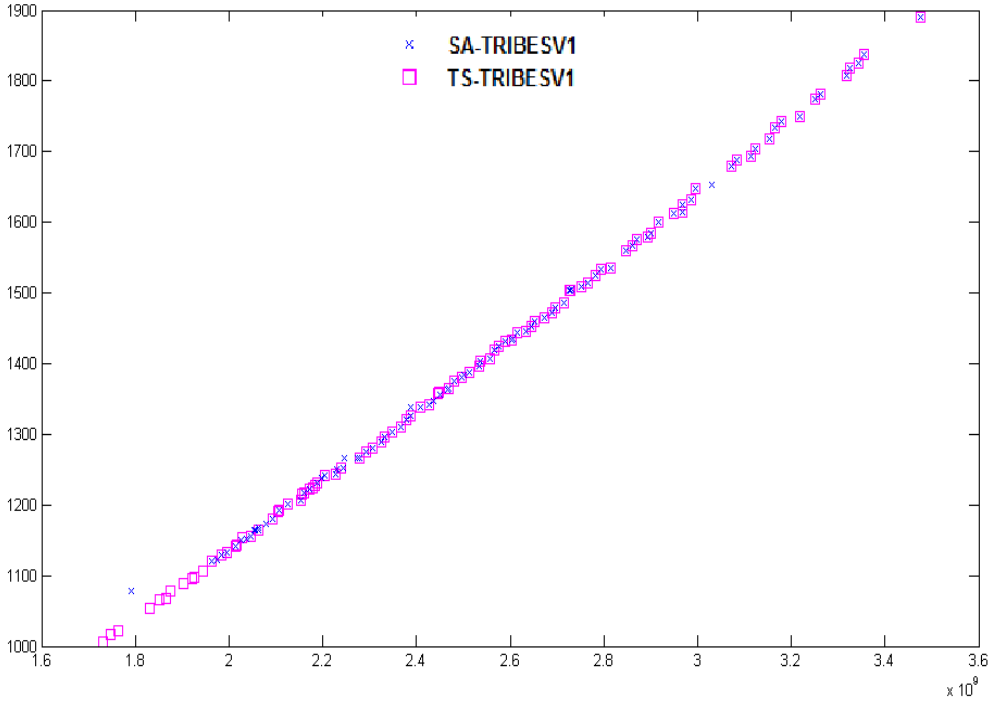


Figure 5.14 : Résultats de l'optimisation du CCII+

Résultats de simulation du convoyeur de courant CCII+ :

La figure 5.14 présente les fronts trouvés par TS-TRIBESV1 et SA-TRIBESV1. Nous constatons que TS-TRIBESV1 donne le meilleur front du point de vue distribution et largeur de front.

Nous présentons, par la suite, les résultats de simulation qui correspondent aux deux solutions optimales présentant les extrémités du front trouvées par l'algorithme TS-TRIBESV1.

Le tableau 5.6 présente les conditions de simulations:

Tableau 5.6 : Conditions de simulation du CCII+

Technologie	0.35µm CMOS AMS
Tension d'alimentation (V_{SS}/V_{DD})	-2,5V/2,5V
Courant de polarisation (I_0)	100µA

Résultats de simulation de la solution 1 :

Tableau 5.7 : Dimensions optimales du CCII+ de la première solution

LN(µm)	*LP(µm)*	*WN(µm)*	*WP(µm)*	f_c*(GHz)*	R_x*(Ω)*
0,59	0,35	2,24	3,88	3,48	1889,99

D'après les figures de simulation, on obtient une résistance R_x égale à 1,9882KΩ et une fréquence f_c égale à 2,3792GHz.

Résultats de simulation de la solution 2 :

Table 5.8 : Dimensions optimales du CCII+ de la deuxième solution

L_N*(µm)*	L_P*(µm)*	W_N*(µm)*	W_P*(µm)*	f_c*(GHz)*	R_x*(Ω)*
0.59	0,35	8	13,86	1 ,73	1006,17

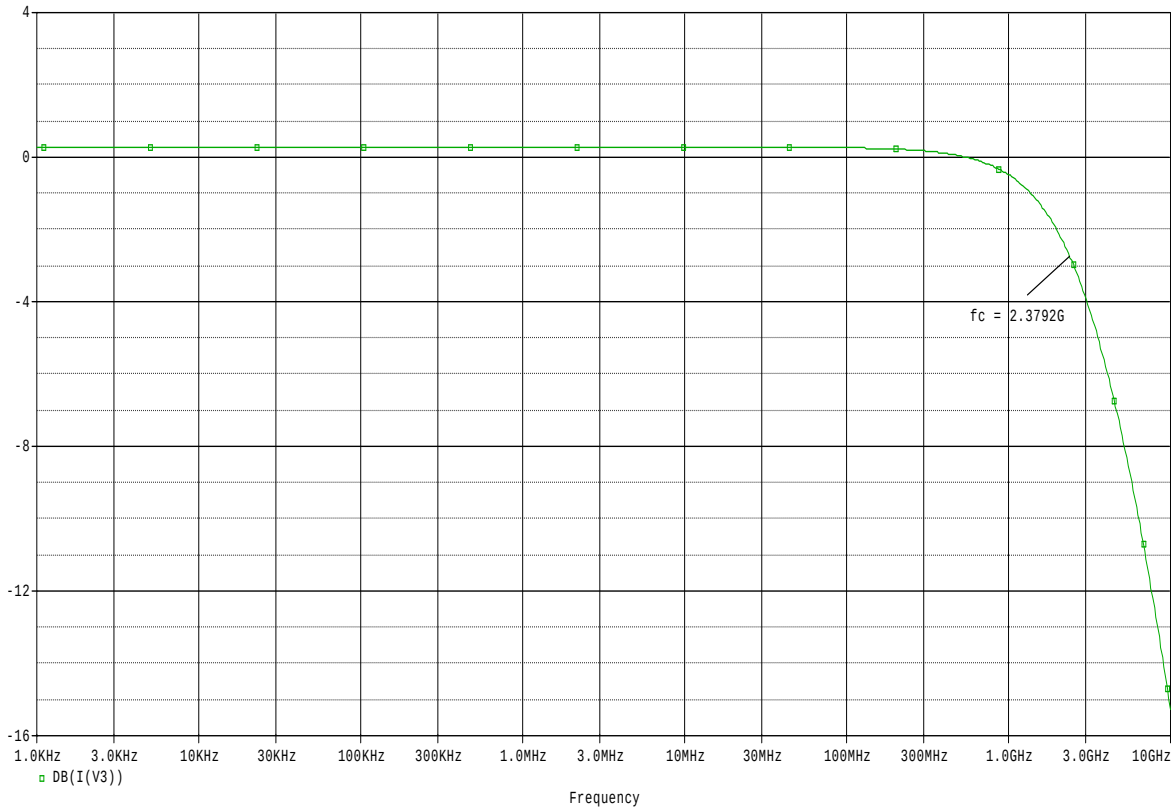


Figure 5.15 : Courbe du gain en courant entre les ports X et Z du CCII+ (Iz/Ix)dB= f(freq)) de la première solution

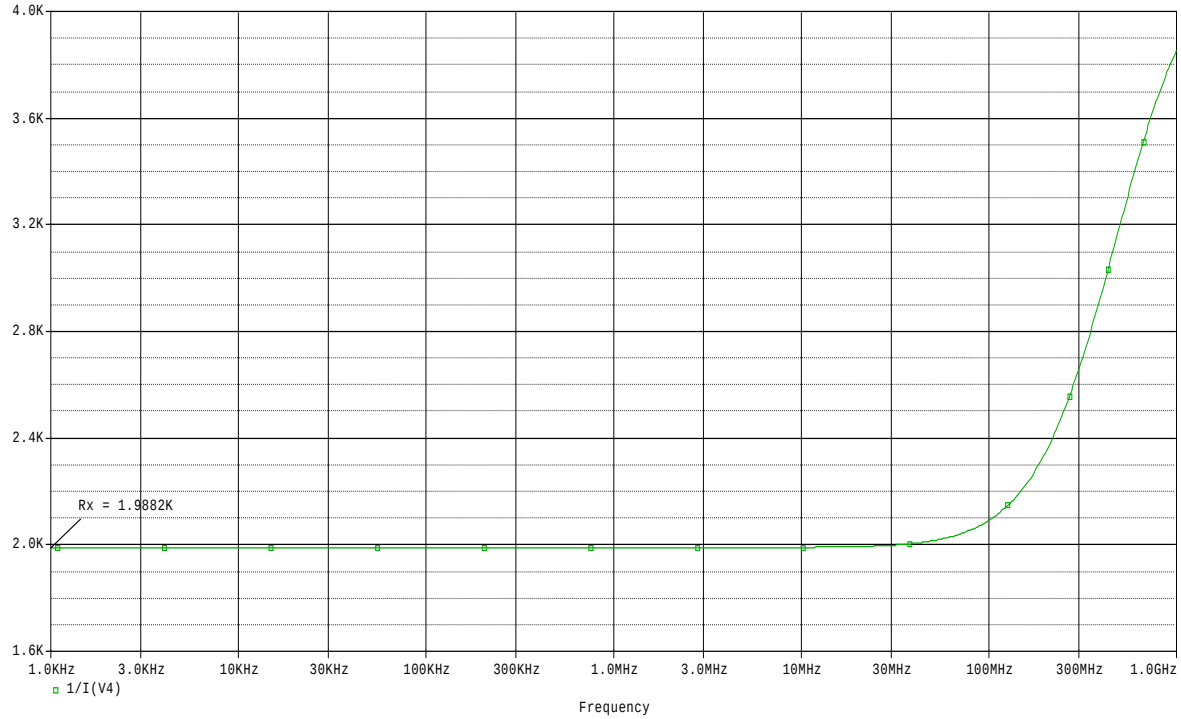


Figure 5.16 : Résultat de simulation de la résistance R_x du CCII+ ($R_x(\Omega)$ = f(freq))

de la première solution

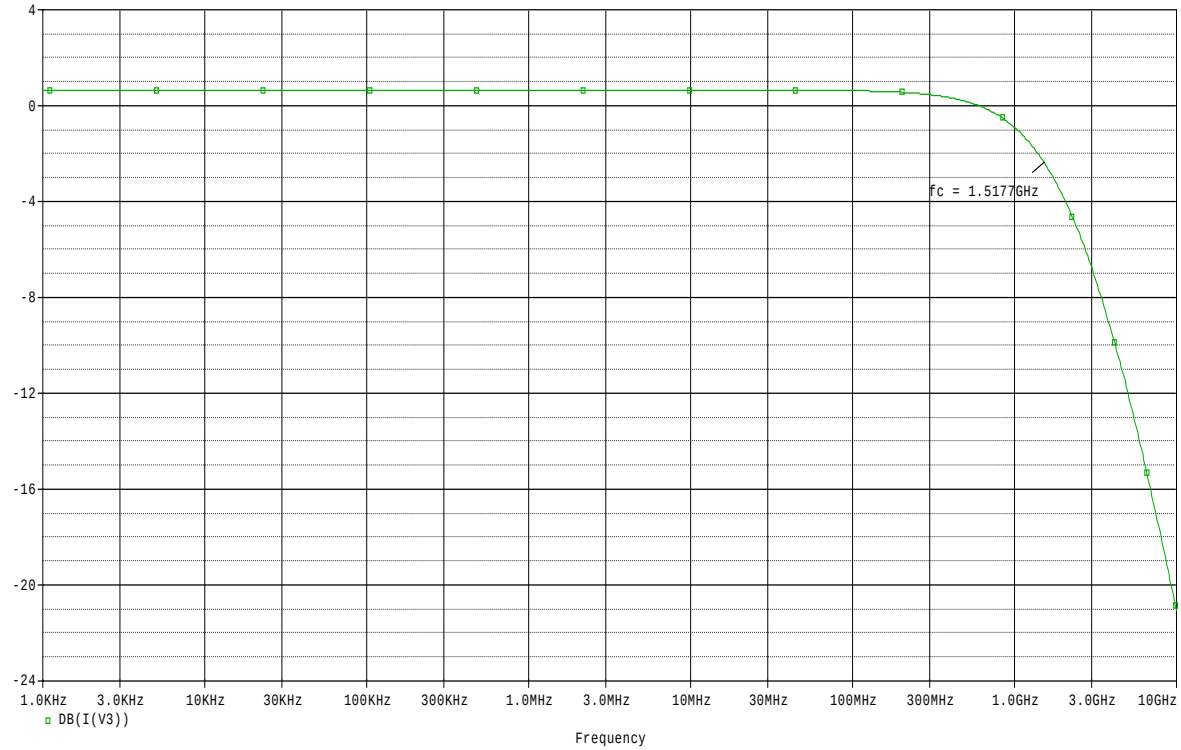


Figure 5.17 : Courbe du gain en courant entre les ports X et Z du CCII+ (Iz/Ix)dB= f(freq))

de la deuxième solution

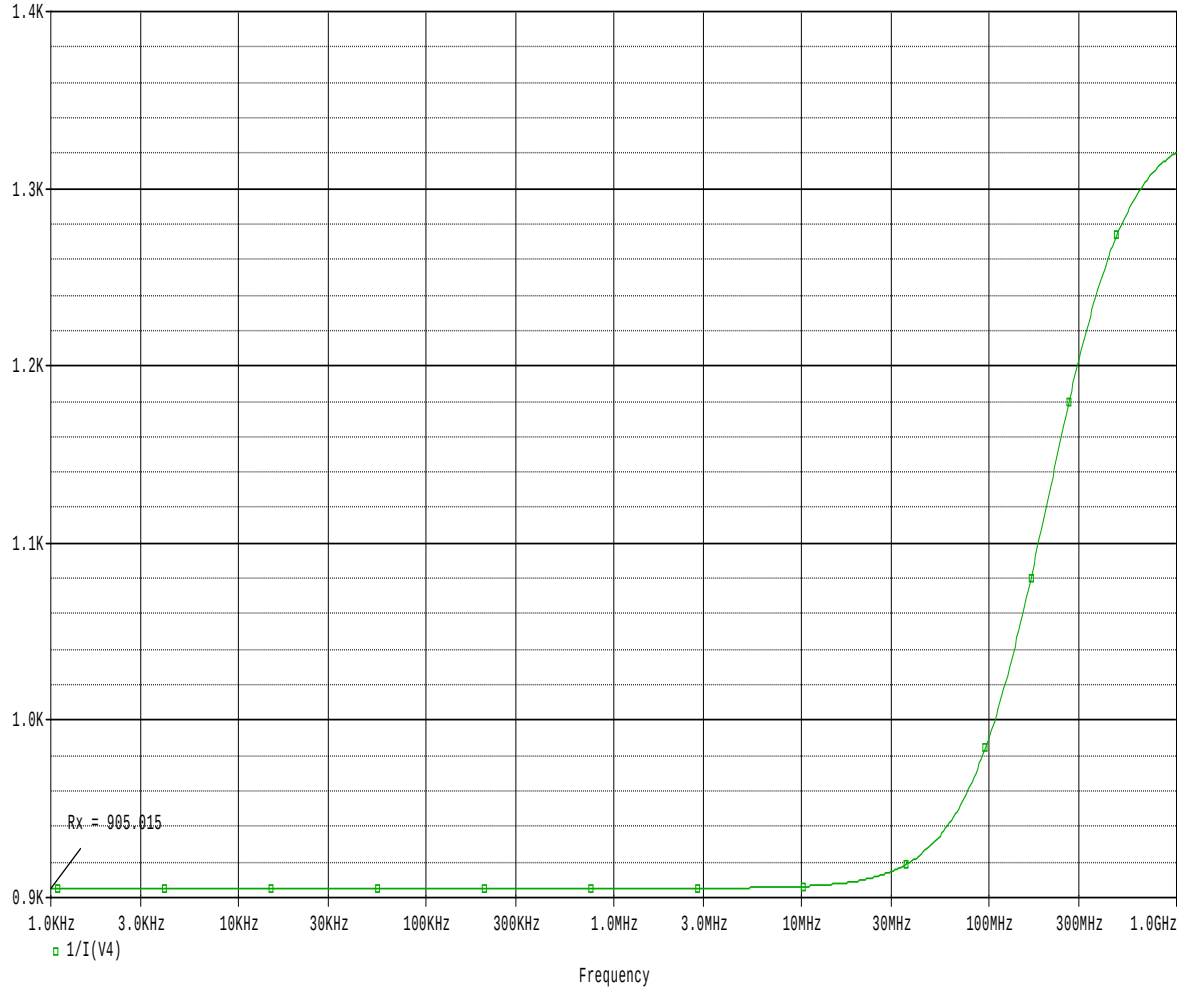


Figure 5.18 : Résultat de simulation de la résistance R_x du CCII+ ($R_x(\Omega) = f(freq)$) de la deuxième solution

Après simulation, nous obtenons une résistance R_x égale à 905,015 Ω et une fréquence égale à 1,5177GHz.

Récapitulation et comparaisons :

Le tableau 5.9 présente une comparaison entre les résultats théoriques et les résultats de simulation pour les deux solutions optimales présentant les extrémités du front de Pareto trouvées par l'algorithme TS-TRIBESV1.

Nous remarquons que les résultats de simulation sont proches des résultats théoriques calculés précédemment. Cela prouve que l'algorithme TS-TRIBESV1 est performant même dans le cas d'une application qui est assez complexe puisqu'elle présente des contraintes.

Tableau 5.9 : Comparaison entre les résultats théoriques et les résultats de simulation du CCII+

	Solution 1		***Solution 2***	
	Optimisation	***Simulation***	***Optimisation***	***Simulation***
f_c ***(Ghz)***	3,48	2,3792	1,73	1,5177
R_x ***(Ω)***	1889,99	1988,2	1006,17	905,015

5.4 Conclusion

Dans ce chapitre, nous avons étudié l'application de TS-TRIBESV1 et SA-TRIBESV1 au problème de dimensionnement des circuits électroniques. Sachant que les performances de ces circuits sont fortement dépendantes des dimensions de leurs transistors, le choix des dimensions est alors une étape déterminante. Réellement, il n'existe pas de méthode automatique pour résoudre ce genre de problèmes. En effet, la résolution est une tâche fastidieuse qui dépend de l'expérience de l'expert. Dans ce chapitre, nous avons montré que l'utilisation des métaheuristiques représente une bonne alternative pour la résolution de ces problèmes tout en assurant un bon compromis entre la qualité des solutions trouvées et le temps de calcul. Pour ce faire, nous avons étudié deux problèmes : le circuit LNA et le convoyeur de courant de deuxième génération. Les résultats trouvés montrent que nos algorithmes sont capables de résoudre efficacement des problèmes assez complexes.

Conclusions et perspectives

Le travail de recherche décrit dans ce mémoire s'est fixé comme objectif d'apporter une contribution à la résolution des problèmes d'optimisation multiobjectif à variables continues. En effet, nous avons présenté un travail basé sur TRIBES, un algorithme adaptatif d'optimisation par essaim particulaire. L'OEP est une métaheuristique basée sur la reproduction des comportements sociaux de certains animaux. L'auto-organisation représente le processus le plus important qui caractérise l'évolution de l'essaim. Il permet l'émergence d'une organisation complexe en partant d'un groupe de particules peu intelligentes. L'inconvénient majeur de l'OEP, comme la plupart des métaheuristiques, est qu'elle est fortement dépendante de son jeu de paramètres. Chaque fois que nous considérons un nouveau problème, nous devons trouver les valeurs idéales pour les différents paramètres. Dans ce cadre se situe l'apport majeur de TRIBES, un algorithme d'OEP qui fonctionne automatiquement, sans paramètres. Il a été mis au point par Maurice Clerc. Notre travail consiste à adapter TRIBES à l'optimisation multiobjectif. L'objectif est d'obtenir un algorithme d'optimisation par essaim particulaire multiobjectif sans paramètres de contrôle.

Nous avons ainsi commencé, en première partie, par définir les concepts de base relatifs à l'optimisation multiobjectif. Nous avons présenté, par la suite, le jeu de fonctions de test et les indicateurs de qualité que nous avons utilisés tout au long de notre travail pour comparer les différents algorithmes considérés.

La seconde partie de ce travail a consisté à étudier et analyser les principales techniques d'optimisation multiobjectif connues dans la littérature. Un intérêt spécial a été accordé aux techniques d'OEP multiobjectif.

La troisième partie a été dévolue à la présentation de notre première adaptation de TRIBES à l'optimisation mutiobjectif. En effet, nous reprenons les principaux mécanismes de TRIBES auxquels sont ajoutés de nouveaux mécanismes destinés à traiter des problèmes multiobjectif. Cette adaptation consiste à utiliser la dominance au sens de Pareto pour respecter l'intégralité de chaque objectif et à ajouter une archive externe pour sauvegarder les solutions non dominées trouvées. Après les expérimentations, nous avons constaté, que TRIBES-Multiobjectif est moins compétitif par rapport à l'algorithme de référence de la littérature à savoir l'algorithme NSGA-II.

La quatrième partie a consisté à ajouter de nouveaux mécanismes pour remédier aux problèmes soulevés précédemment. Dans un premier temps, nous avons proposé l'hybridation entre TRIBES-Multiobjectif et un algorithme de recherche locale. L'idée était d'améliorer la capacité d'exploitation de TRIBES-Multiobjectif. En fait, la recherche locale ne va pas être nécessairement appliquée d'une manière canonique, c'est-à-dire sur toutes les particules de l'essaim (LS-TRIBESV3) : nous avons envisagé deux autres manières, à savoir l'appliquer uniquement sur la meilleure particule de chaque tribu (LS-TRIBESV2) ou bien aussi sur les particules de l'archive (LS-TRIBESV1). Par la suite, nous avons étudié son hybridation avec la recherche tabou et le recuit simulé. Les résultats trouvés ont montré que l'hybridation est fructueuse et que LS-TRIBESV1 donne les meilleures performances.

La dernière partie a été consacrée à l'application de nos algorithmes dans le domaine de l'électronique. En effet, nous avons considéré le problème du dimensionnement des transistors dans les circuits analogiques. Peu de méthodes automatiques existent dans la littérature. L'application de TRIBES à ce genre de problème apparait comme une bonne alternative pour les concepteurs de ces circuits puisqu'il se caractérise par une prise en main facile et ne nécessite aucun réglage au préalable.

Les perspectives de nos travaux sont multiples et concernent en particulier :

- Adopter de nouvelles stratégies d'adaptation comportementales et structurelles. En effet, concernant les stratégies de déplacement, les stratégies actuelles sont toujours en fonction des positions des particules informatrices et du guide personnel. Nous pouvons envisager à ce niveau d'élargir cet ensemble en prenant en considération de nouveaux informateurs choisis soigneusement à partir de l'archive et de l'essaim. Ce choix doit être effectué de manière à promouvoir la diversité dans l'essaim. Concernant les stratégies d'adaptation structurelle, il s'avère aussi nécessaire de trouver un moyen de détecter le moment où l'essaim commence à stagner. Dans ce cas, chaque fois que nous aurons une stagnation, nous pourrons envisager de réinitialiser tout l'essaim.
- Concernant les schémas d'hybridation, nous pouvons envisager d'autres schémas d'hybridation qui se contentent d'emprunter certains mécanismes relatifs à d'autres algorithmes et l'ajouter au notre. Nous pouvons proposer à ce niveau de nouvelles stratégies pour la sélection des guides qui se basent sur les méthodes de sélection utilisées dans les algorithmes génétiques.
- Ajouter de nouvelles stratégies pour l'approximation et l'héritage de la fonction objectif. En effet, l'évaluation de la fonction objectif est souvent couteuse de point de

vue du temps d’exécution. Dans ce cas, l’idée est d’appliquer des stratégies d’approximation ou d’héritage pour des particules bien choisies afin de diminuer la complexité de TRIBES tout en gardant les mêmes performances.

Références bibliographiques

[Abido, 07] M.A. Abido. Two-Level of Nondominated Solutions Approach to Multiobjective Particle Swarm Optimization. GECCO'07, July 7–11, 2007, London, England, United Kingdom, pp. 726-733.

[Alvarez-Benitez et al., 05] J.E. Alvarez-Benitez, R.M. Everson & J.E. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In C. A. Coello Coello et al., editor, EMO 2005, number 3410 in LNCS, pp. 459–473.

[Andreani et al., 01] P. Andreani & H. Sjoland, Noise optimization of an inductively degenerated CMOS low noise amplifier. The IEEE Transactions on Circuits and Systems, 2001, Vol. 48, pp. 835–841.

[Angeline, 98] P. Angeline. Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences. Proceedings of the 7th Annual Conference on Evolutionary Programming, San Diego, California, USA, March 1998, pp. 601-610.

[van den Berg, 02] F. van den Bergh. An Analysis of Particle Swarm Optimizers. PhD thesis, Departement of Computer Science, University of Pretoria, Pretoria , South Africa, November 2002.

[Berro, 01] A. Berro. Optimisation Multiobjectif et Stratégies d'Evolution en Environnement Dynamique. Thèse de Doctorat, Université des Sciences Sociales Toulouse 1, Décembre 2001.

[Bartz-Beielstein et al., 03] T. Bartz-Beielstein, P. Limbourg, K.E. Parsopoulos, M.N. Vrahatis, J. Mehnen & K. Shmitt. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In congress on Evolutionary Computation (CEC'2003), Canberra, Australia, December 2003, IEEE Press, Vol. 3, pp. 1780-1787.

[Boughariou et al., 10] M. Boughariou, M. Fakhfakh & M. Loulou. Design and Optimization of LNAs through the Scattering Parameters. The IEEE Mediterranean Electrotechnical Conference (MELECON), Valletta, Malta, April 2010.

[Cabrera et al., 10] J.C.F. Cabrera & C.A.C. Coello. Micro-MOPSO: A Multi-Objective Particle Swarm Optimizer that Uses a Very Small Population Size. in Nadia Nedjah, Leandro dos Santos Coelho and Luiza de Macedo de Mourelle (editors), Multi-Objective Swarm Intelligent Systems. Theory & Experiences, Springer, Berlin Heidelberg, 2010, ISBN 978-3-642-05164-7, chapter 4, pp. 83-104.

[Carlos et al., 00] A. Carlos & C.A.C. Coello. An Updated Survey of GA-Based Multiobjective Optimization Techniques. ACM Computing Surveys, June 2000, Vol. 32, No. 2, pp. 109-143.

[Chelouah et al., 00] R. Chelouah & P. Siarry. Tabu Search applied to global optimization. European Journal of Operational Research 123, 2000, pp. 256-270.

[Clerc et al., 02] M. Clerc & J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, 2002, 6: pp. 58–73.

[Clerc, 99] M. Clerc. The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. Proceedings of the Congress on Evolutionary Computation, Washington DC, USA, July 1999, pp. 1951- 1957.

[Clerc, 03] M. Clerc. TRIBES-Un exemple d'optimisation par essaim particulaire sans paramètres de contrôle. Conférence OEP'03, 2 Octobre , 2003, Paris, France.

[Clerc, 06] M. Clerc. Particle Swarm Optimization. International Scientific and Technical Encyclopaedia, John Wiley & sons, 2006.

[Coello, 99] C.A.C. Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. Knowledge and Information Systems, Vol. 1, No. 3, 1999, pp. 269-308.

[Coello et al., 02] C.A.C. Coello & M.S. Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. Congress on Evolutionary Computation (CEC'2002), IEEE Service Center, Piscataway, New Jersey, May 2002, Vol. 2, pp. 1051-1056.

[Coello et al., 02] C.A.C. Coello & M.S. Lechuga. MOPSO: A proposal for multiobjective particle swarm optimization. In Congress on Evolutionary Computation (CEC'2002), volume 2, Piscataway, New Jersey, May 2002. IEEE service center, pp. 1051-1056.

[Colette et al., 02] Y. Colette & P. Siarry. Optimisation Multiobjectif. Eyrolles, 2002.

[Cooren, 09] Y. Cooren. Perfectionnement d'un algorithme adaptatif d'optimisation par essaim particulaire. Applications en génie médical et en électronique. PhD thesis, Université Paris 12, November 2009.

[Coello et al., 04] C.A.C. Coello, G.T. Pulido & M.S. Lechuga. Handling multiple objectives with particle swarm optimization. IEEE Transactions on Evolutionary Computation, June 2004, 8(3): pp. 256-279.

[Deb et al., 02] K. Deb, S. Agrawal, A. Pratap & T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[Deb et al., 02] K. Deb, S. Agrawal, A. Pratap & T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multiobjective Optimization : NSGA-II. IEEE Transactions on Evolutionary Computation, 2002, Vol. 5, N°3, pp. 115-148.

[Durillo et al., 09] J.J. Durillo, J. García-Nieto, A.J. Nebro, C.A.C. Coello, F. Luna, and E. Alba. Multi-Objective Particle Swarm Optimizers: An Experimental Comparison. 5th International Conference, EMO 2009, Nantes, France, April 2009, pp.495-509.

[Ehrgott et al., 00] M. Ehrgott & X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spektrum, 2000, Vol. 22, pp.425-460.

[Fakhfakh et al., 10] M. Fakhfakh & M. Loulou. Live demonstration: CASCADES. 1: A flowgraph-based symbolic analyzer. The IEEE International Symposium on Circuits and Systems (ISCAS), Paris, France, June 2010, pp. 2782.

[Fogel, 00] D. Fogel. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence (second edition). IEEE Press, 2000.

[Fonseca et al., 93] C.M. Fonseca & P.J. Fleming. Genetic Algorithm for Multiobjective Optimization: Formulation, Discussion, Generalization. In Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, October 1993, pp. 416-423.

[Fourman, 85] M.P. Fourman. Compaction of Symbolic Layout Using Genetic Algorithms. In Genetic Algorithms and their Applications. Proceedings of the First International Conference on Genetic Algorithm, Pittsburgh, PA, USA, July 1985, pp. 141-153.

[Glover et al., 97] F. Glover & M. Laguna. Tabu search. Kluwer Academic Publishers, 1997.

[Goldberg, 89] D.E. Goldberg. Genetic algorithms for search, optimization, and machine learning. Reading, MA: Addison-Wesley, 1989.

[Huang et al., 07] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J .Liang, M. Preuss & S. Huband. Problem Definitions for Performance Assessment & Competition on Multi-objective Optimization Algorithms. Nanyang Technological University, Singapore, Tech. Rep, 2007.

[Huband et al., 06] S. Huband, P. Hingston, L.Barone & L.While. A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit. IEEE Transactions on Evolutionary Computation, 2006, 10(5), pp. 477-506.

[Hu et al., 02] X. Hu & R. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In Congress on Evolutionary Computation (CEC'2002), Piscataway, New Jersey, May 2002, IEEE Service Center, Vol. 2, pp. 1677-1681.

[Hu et al., 03] X. Hu, R. Eberhart & Y.Shi. Particle swarm with Extended Memory for multiobjective Optimization. In IEEE Swarm Intelligence Symposium, Indianapolis, IN, USA, April 2003, pp. 193-197.

[Kennedy et al., 95] J. Kennedy & R.C. Eberhart. Particle Swarm Optimisation. Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, November-December 1995, IEEE Press, pp. 1942-1948.

[Kennedy, 97] J. Kennedy. The Particle Swarm: Social Adaptation of Knowledge. Proceedings of the 1997 IEEE international Conference on Evolutionary Computation ICEC'97, Indianapolis, Indiana, USA, April 1997, pp. 303-308.

[Kennedy, 03] J. Kennedy. Bare bones particle swarms. Proceedings of the 2003 IEEE Swarm Intelligence Symposium, University Place Conference Center, Indianapolis, Indiana, USA, April 2003, IEEE Press, pp. 80-87.

[Kirkpatrick et al., 83] S. Kirkpatrick, C.D. Gelatt & M.P. Vecchi. Optimization by Simulated Annealing. Science, 1983, Vol. 220, pp. 671-680.

[Knowles et al., 00] J.D. Knowles & D.W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation, 2000, 8(2), pp. 149– 172.

[Knowles et al., 06] J. Knowles, L.Thiele & E.Zitler. *A tutorial on the Performance Assessement of Stochastic Multi-objective Optimizers*. Tik-Report No-214, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, Feb 2006.

[Li, 04] X. Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In Erick Cantù-Paz et al., editor, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2004), Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102, pp. 117-128.

[Li et al., 07] X.Li & A.P. Engelbrecht. Particle Swarm Optimization: an introduction and its recent developments. GECCO'07, July 7–11, 2007, London, England, United Kingdom. ACM 978-1-59593-698-1/07/0007.

[Moore et al., 99] J. Moore & R. Chapman. Application of particle swarm to multiobjective optimization. Technical report, Departement of Computer Science and Software Engineering, Auburn University, 1999.

[Mostaghim et al., 03] S. Mostaghim & J. Teich. Strategies for finding good local guides in multi- objective particle swarm optimization (MOPSO). In Proceedings of the IEEE Swarm Intelligence Symposium, SIS '03, Indianapolis, Indiana, USA, April 2003, pp.26–33.

[Niu et al., 05] B. Niu, Y. Zhu, X. He & W. Henry. MCPSO : A multi-swarm cooperative particle swarm optimizer. Applied Mathematics and Computation, 2005, Vol. 185, N° 2, pp. 1050-1062.

[Okabe et al., 04] T. Okabe, Y. Jin, M. Olhofer & B. Sendhoff. On Test Functions for Evolutionary Multi-objective Optimization. In Parallel Problem Solving from Nature (PPSN VIII), LNCS 3242, Springer, 20004, pp.792–802.

[Osyczka, 85] A. Osyczka. Multicriteria optimization for engineering design. In John S. Gero, editor, Design Optimization, Academic Press, 1985, pp.193–227.

[Ozcan et al., 98] E. Ozcan & C.K. Mohan. Analysis of a simple particle swarm optimization system. In intelligent Engineering Systems Through Artificial Neural Networks, 1998, Vol. 8, pp. 253-258.

[Parsopoulos et al., 02] K.E. Parsopoulos & M.N. Vrahatis. Particle Swarm Optimization Method in Multiobjective Problems. Proceedings of the ACM 2002 Symposium on Applied Computing (SAC'2002), Madrid, Spain, March 2002, pp. 603-607.

[Parsopoulos et al., 04] K.E. Parsopoulos, D.K. Tasoulis & M.N. Vrahatis. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004), Innsbruck, Austria, February 2004, ACTA Press, Vol. 2, pp. 823-828.

[Pulido et al., 04] G.T. Pulido & C.A.C. Coello. Using clustering techniques to improve the performance of a particle swarm optimizer. In Kalyanmoy Deb et al., editor, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2004), Seattle, Washington, USA, June 2004, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102, pp. 225-237.

[Pulido, 05] G.T. Pulido. On the Use of Self-Adaptation and Elitism for Multiobjective Particle Swarm Optimization. PhD Thesis, Department of Electrical Engineering, Computer Science option, September 2005.

[Quintero et al., 08] L.V.S. Quintero, N.R. Santiago & C.A.C. Coello. Towards a More efficent Multiobjective Particle Swarm Optimizer. In Lam Thu Bui and Sameer Alam (editors), Multiobjective Optimization in computational intelligence: Theory and practice, 2008, Information Science Reference, Hershey, USA, pp.76-105.

[Ray et al., 02] T. Ray & K.M. Liew. A swarm metaphor for multiobjective design optimization. Engineering Optimization, March 2002, 34(2): pp. 142-153.

[Schaffer, 84] J.D. Schaffer. Some Experiments in Machine Learning using Vector Evaluated Genetic Algorithms. PhD Thesis, Vanderbilt University, December 1984.

[Schmid, 00] H.Schmid. Approximating the Universal Active Element. IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, 2000, Vol. 47, N° 11.

[Schwefel, 81] H.P. Schwefel. Numerical optimization of computer models. Wiley, Chichester, 1981.

[Seevinck, 00] E. Seevinck. CMOS Translinear Circuits for Minimum Supply Voltage. IEEE Transactions on Circuits and Systems-II : Analog and Digital Signal Processing, 2000, Vol. 47, N°12, pp. 1560_1564.

[Serra et al., 97] P. Serra, A.F. Stanton & S. Kais. Method for global optimization. Physical Review, 1997, Vol. 55, pp. 1162-1165.

[Shi et al., 98] Y. Shi & R. Eberhart. Parameter Selection in Particle Swarm Optimization. Proceedings of the 7th Annual Conference on Evolutionary Programming, Berlin, Germany, March 1998, pp.591-600.

[Siarry et al., 97] P. Siarry & G. Berthiau. Fitting Of Tabu Search To Optimize Functions Of Continuous Variables. International Journal For Numerical Methods In Engineering, 1997, Vol. 40, pp. 2449-2457.

[Sierra et al., 05] M.R. Sierra & C.A.C. Coello. Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance. In third International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005, Guanajuata, Mexico, 2005, LNCS 3410, Springer-verlag, pp. 505-519.

[Sierra et al., 06] M.R. Sierra & C.A.C. Coello. Multiobjective Particle Swarm Optimizers: A survey of the State-Of-the-Art. International Journal of Computational Intelligence Research, 1987, Computer Graphics, Vol. 21, N°4, pp. 25-34.

[Sierra et al., 07] M.R. Sierra & C.A.C. Coello. A study of techniques to improve the efficiency of a multi-objective particle swarm optimizer. Evolutionary Computation in Dynamic and Uncertain Environments, 2007, Springer, pp. 269-296.

[Smairi et al., 10] N. Smairi, S. Bouamama, K. Ghedira & P. Siarry. New Proposal For a Multi-objective Technique Using TRIBES and Tabu Search. Proceedings of the

7th International Conference on Informatics in Control, Automation and Robotics, Madeira, Portugal, June 2010, Vol. 1, pp. 86-91.

[Smairi et al., 10b] N. Smairi, S. Bouamama, K. Ghedira & P. Siarry. A Study of the Efficiency of the Hybridization of the Particle Swarm Optimizer and Tabu Search. KES'10, September, 2010, London, England, United Kingdom.

[Srivinas et al., 95] N. Srivinas & K. Deb. Multiobjective Function Optimization Using Nondominated Sorting Genetic Algorithms. Evol. Comput., 1995, Vol. 2, N°3, pp. 221-248.

[Zitzler et al., 07] E. Zitzler & K. Deb. Tutorial on Evolutionary Multiobjective Optimization. GECCO'07, July 7–11, 2007, London, United Kingdom. ACM 978-1-59593-698-1/07/0007.

[Zitzler et al., 98] E. Zitzler & L. Thiele. Multiobjective optimization using evolutionary algorithms: a comparative case study. In Conference on Parallel Problem Solving from Nature (PPSN V), Amsterdam, The Netherlands, September 1998, pp. 292–301.

[Zielinski et al., 07] K. Zielinski & R. Laur. Adaptive Parameter Setting for a Multi-Objective Particle Swarm Optimization Algorithm. Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, September 2007, IEEE Press, pp. 3019 - 3026.