



HAL
open science

Les modèles de substitution pour l'optimisation rapide des circuits analogiques

Nawel Drira

► **To cite this version:**

Nawel Drira. Les modèles de substitution pour l'optimisation rapide des circuits analogiques. Automatique. Université Paris-Est; École nationale d'ingénieurs de Gabès (Tunisie), 2019. Français. NNT : 2019PESC0070 . tel-03507220

HAL Id: tel-03507220

<https://theses.hal.science/tel-03507220>

Submitted on 3 Jan 2022

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

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



THÈSE EN COTUTELLE

pour obtenir le grade de

DOCTEUR DE L'ÉCOLE NATIONALE D'INGÉNIEURS DE GABÈS

Discipline : Génie Électrique

Et

DOCTEUR DE L'UNIVERSITÉ PARIS-EST

École doctorale de Mathématiques et STIC (MSTIC, ED 532)

Discipline : Signal, Image, et Automatique (CNU 61)

Thèse présentée par

Nawel DRIRA

le 17 décembre 2019

Les modèles de substitution pour l'optimisation rapide des circuits analogiques

Thèse dirigée par Pr. Patrick SIARRY et Pr. Mourad FAKHFAKH

Composition du Jury :

M. Gérard BERTHIAU	Professeur à l'Université de Nantes, France	Rapporteur
M. Néjib HASSEN	Professeur à l'Université de Monastir, Tunisie	Rapporteur
M. Mohamed Naceur ABDELKRIM	Professeur à l'Université de Gabès, Tunisie	Examineur
Mme. Edwige PISSALOUX	Professeur à l'Université de Rouen, France	Examinatrice
M. Mourad FAKHFAKH	Professeur à l'Université de Sfax, Tunisie	Directeur de la thèse
M. Patrick SIARRY	Professeur à l'Université Paris-Est, Créteil, France	Directeur de la thèse

Résumé

La conception, la synthèse, le dimensionnement et l'optimisation des circuits analogiques sont devenus trop compliqués pour qu'ils soient traités par les méthodes classiques de dimensionnement. Deux techniques sont largement utilisées par les concepteurs pour développer des outils d'aide à la conception, à savoir, la technique d'optimisation basée sur les équations symboliques et la technique d'optimisation basée sur les simulations (connue sous le nom de '*inloop technique*'). L'optimisation basée sur les équations est une technique rapide, cependant vu que les fonctions de transfert manipulées sont a fortiori entachées d'erreurs (principalement à cause des composants non-linéaires), ses performances restent limitées surtout pour les circuits complexes. Par contre, l'optimisation basée sur les simulations est très précise, mais en contrepartie elle nécessite de longues durées d'exécution.

Dans ce travail, on s'intéresse à l'adaptation de la technique de métamodélisation pour l'analyse des circuits analogiques. La métamodélisation est une technique qui combine les avantages des deux approches de dimensionnement/optimisation citées plus haut. Un intérêt particulier est porté à la technique EGO (*Efficient Global Optimization*), récemment proposée dans la littérature, vu que cette dernière associe intrinsèquement « modélisation » et « optimisation », ajouté à ceci le fait qu'elle permet aussi la génération de modèles précis, tout en utilisant des bases de données de tailles réduites. C'est ainsi que l'on a étudié de près cette technique et qu'on l'a adaptée pour maximiser les performances de circuits analogiques. Des études comparatives ont été menées pour juger de l'efficacité de cette technique tout en la comparant aux approches classiques.

Des améliorations de la technique EGO ont été proposées, à savoir l'intégration dans l'algorithme correspondant du critère '*pseudo expected improvement*' pour développer un EGO 'parallèle'. Une seconde amélioration a aussi été proposée dans ce travail et qui consiste en la transformation de la technique EGO, de nature mono-objectif, en une approche multiobjectif en utilisant le critère '*expected improvement matrix*'.

Toutes les approches proposées ont été validées par application, d'abord sur des fonctions mathématiques de test, puis sur des circuits analogiques. Les résultats obtenus ont été comparés à ceux obtenus par simulation. Des métriques statistiques ont été utilisées à cet effet. Les avantages des approches proposées ont été ainsi mis en relief.

Mots-clés : dimensionnement des circuits analogiques, optimisation basée sur les simulations, optimisation basée sur les équations, métaheuristiques, métamodélisation, *krigeage*.

Abstract

Design, synthesis, sizing, and optimization of analog circuits have become too complex to be processed by classical sizing methods. There exist two commonly used techniques to develop tools that facilitate the design, known as the equation-based optimization, and simulation-based optimization (also called ‘in-loop’) approaches. The equation-based approaches are more rapid yet less precise; their performance is limited, especially for complex circuits. On the other hand, the simulation-based techniques are highly precise; nevertheless, they are more costly in terms of computational time. In this thesis, we focus on the adaptation of the metamodeling technique for the analysis of analog circuits.

Metamodeling is a technique that combines the advantages of the two sizing/optimization approaches mentioned above. Recently, the EGO (Efficient Global Optimization) technique been proposed. It is gaining attention of analog designers, as it essentially combines "modeling" and "optimization". Furthermore, it allows the generation of precise models as well, while using small-size databases. We studied this technique and adapted it to maximize performances of analog circuits. Comprehensive studies have been conducted to evaluate the effectiveness of this technique and compare it to classical approaches. In our work, we propose two improvements for the EGO technique; the first consists of integrating the algorithm with the “pseudo expected improvement” criterion in order to parallelize EGO. The second is about transforming the mono-objective EGO technique into a multi-objective one by using the “expected improvement matrix” criterion.

All proposed approaches were evaluated by application, firstly through mathematical test functions, and then on analog circuits. Obtained results were compared with those obtained by simulation. Statistical metrics have been used for this purpose. The benefits of the proposed approaches have been highlighted.

Key-words: design of analog circuits, equation-based approach, simulation-based (in-loop based) approach, metaheuristics, metamodeling, kriging, efficient global optimization, expected improvement, expected improvement matrix, pseudo expected improvement.

Table des matières

Introduction générale.....	1
Chapitre 1 : Métaheuristiques et métamodélisation : état de l’art	3
1.1. Introduction.....	3
1.2. Les techniques d’optimisation.....	3
1.2.1. L’optimisation mono-objectif.....	4
1.2.1.1. Définition du problème.....	4
1.2.1.2. Les algorithmes évolutionnaires	5
1.2.1.3. L’optimisation par essais particuliers	9
1.2.2. L’optimisation multiobjectif.....	12
1.2.2.1. Définition du problème.....	12
1.2.2.2. Relations d’ordre et de dominance.....	13
1.2.2.3. Frontière de Pareto.....	13
1.2.2.4. Métaheuristiques pour l’optimisation multiobjectif	14
1.3. Application des métaheuristiques aux problèmes de dimensionnement des circuits analogiques.....	16
1.3.1. La technique d’optimisation basée sur les équations	18
1.3.2. La technique d’optimisation basée sur la simulation	18
1.4. Les modèles de substitution	20
1.4.1. Définition	21
1.4.2. Les plans d’expériences	21
1.4.2.1. Les plans d’expériences classiques	22
1.4.2.2. Les plans hypercubes latins	22
1.4.3. Les principales techniques de métamodélisation.....	23
1.4.3.1. Le modèle polynomial.....	23
1.4.3.2. Les fonctions à bases radiales	23
1.4.3.3. La technique de <i>krigeage</i>	24
1.5. Conclusion.....	24
Chapitre 2. La technique d’optimisation basée sur les modèles de substitution- <i>Efficient Global Optimization</i> (EGO)	26
2.1. Introduction.....	26

2.2. L'algorithme EGO.....	26
2.2.1. Le critère d'amélioration espérée.....	27
2.3. Validation et comparaison avec des fonctions mathématiques.....	29
2.3.1. La métrique "Wilcoxon Signed Rank Test".....	30
2.3.2. Résultats	32
2.4. Application et comparaison sur un circuit analogique.....	34
2.4.1. Optimisation des performances d'un amplificateur opérationnel à transconductance (OTA).....	34
2.4.1.1. Maximisation du gain	35
2.4.1.2. Maximisation du taux de rejection du mode commun	36
2.4.1.3. Maximisation du taux de rejection du bruit d'alimentation	38
2.4.2. Récapitulatif et interprétations.....	39
2.5. Conclusion.....	39
Chapitre 3 : Amélioration de l'algorithme EGO : rapidité et précision.....	40
3.1. Introduction.....	40
3.2. Influence de la taille de la base de données.....	40
3.2.1. Application sur des circuits analogiques	41
3.2.1.1. Optimisation des performances du circuit convoyeur de courant	41
3.2.1.2. Optimisation des performances d'un suiveur de tension	44
3.2.2. Évaluation et interprétation.....	46
3.3. Le critère d'amélioration 'Pseudo expected improvement' (PEI).....	51
3.3.1. Principe de fonctionnement de l'algorithme EGO parallèle.....	51
3.3.2. Le critère d'amélioration PEI.....	51
3.3.3. Application sur des fonctions tests et comparaison avec l'algorithme EGO conventionnel.....	53
3.3.4. Application à des circuits analogiques	54
3.3.4.1. Cas du convoyeur de courant	54
3.3.4.2. Cas du suiveur de tension.....	57
3.3.5. Récapitulatif et interprétations.....	58
3.4. Adaptation de l'algorithme EGO pour l'optimisation des performances multiobjectif...58	
3.4.1. Principe de fonctionnement de l'algorithme EGO multiobjectif.....	59
3.4.2. Le critère d'amélioration EIM.....	59
3.4.3. Application sur des fonctions tests.....	61

3.4.4. Application sur un circuit analogique.....	62
3.5. Conclusion.....	63
Conclusion générale et perspectives	64
Annexe : les fonctions de test.....	66
Références bibliographiques	74

Introduction générale

L'augmentation de la complexité de la tâche de conception des circuits analogiques, l'effet des variations des technologies et la demande de réduction du temps de conception, sont des défis auxquels les concepteurs font face. Ces difficultés ont augmenté la nécessité de développer de nouvelles méthodologies en vue de l'automatisation de la conception des circuits analogiques. Par conséquent, la tâche d'optimisation est devenue inéluctable. Des méthodes de dimensionnement/optimisation sont basées sur des boucles itératives entre un algorithme d'optimisation et les évaluateurs des performances [Barros *et al.*, 2010]. Deux principales techniques d'évaluation des performances sont généralement utilisées. La première méthode, basée sur des équations analytiques, est rapide, mais inexacte [Sallem, 2015] [Fakhfakh *et al.*, 2010] [Huelsman, 1996]. À l'inverse, la deuxième, qui est basée sur la simulation, est précise, mais lente [Sallem *et al.*, 2009] [Barros *et al.*, 2010] [Sawal Hamid, 2009].

Ces dernières années, les concepteurs ont recouru à une nouvelle technique de dimensionnement/optimisation pour réduire le temps de cycle de conception des circuits analogiques. Cette technique est basée sur l'utilisation des modèles de substitution. Ces derniers ont été récemment proposés dans la littérature [Simpson *et al.*, 1998] [Jin *et al.*, 2001] [Forrester *et al.*, 2008]. Nous distinguons plusieurs méthodes de métamodélisation. Parmi lesquelles, nous citons : les méthodes polynomiales [Forrester *et al.*, 2008], les techniques de *krigeage* [J. Sack *et al.*, 1989], les fonctions à base radiale [W.S. Simpson *et al.*, 2001]. Ces modèles de substitution permettent d'établir des modèles précis, avec un temps d'évaluation très court.

À la fin des années 1990, l'algorithme « *Efficient Global Optimization* » (EGO) a été proposé par [Jones *et al.*, 1998]. Il a été, spécialement, adopté afin de diminuer le nombre d'appels aux fonctions coûteuses.

Cet algorithme associe intrinsèquement « modélisation » et « optimisation ». La fonction à minimiser (ou à maximiser) est interpolée de manière probabiliste par la technique de *krigeage*. Cette technique permet d'ajouter des points d'amélioration à ce modèle pendant le processus d'optimisation à travers le critère « *Expected Improvement* » (EI). L'algorithme d'évolution différentielle (DE) est utilisé pour maximiser le critère d'amélioration espérée (EI). Ce critère a pour objectif d'évaluer et de mettre à jour le modèle de *krigeage* établi [Zhan *et al.*, 2017].

Dans ce travail de thèse, nous nous intéressons à l'adaptation et l'application de l'algorithme EGO aux problèmes de dimensionnement des circuits analogiques afin de réduire la durée du

cycle de conception. Nous proposons aussi deux améliorations pour cette approche. La première est l'intégration dans l'algorithme correspondant du critère '*pseudo expected improvement*' pour développer un EGO 'parallèle'. Une seconde amélioration proposée dans ce travail consiste en la transformation de la technique EGO, de nature mono-objectif, en une approche multiobjectif, en utilisant le critère '*expected improvement matrix*'.

Ce rapport est structuré en trois chapitres. Le premier chapitre est consacré à l'étude bibliographique des méthodes d'optimisation mono-objectif et multiobjectif. Un intérêt particulier est accordé aux métaheuristiques évolutionnaires et aux processus d'évaluation et de dimensionnement des circuits analogiques. Aussi, nous présentons les techniques des modèles de substitution les plus utilisées dans la littérature et les étapes qu'il faut respecter pour générer un modèle précis.

Dans le deuxième chapitre, nous présentons l'efficacité de l'algorithme EGO. Au début, nous appliquons cette approche sur des fonctions mathématiques et nous la comparons à deux métaheuristiques (PSO et GA) basées sur la technique de "*krigeage*" en utilisant la métrique « *Wilcoxon Signed Rank Test* ». Ensuite, nous proposons d'appliquer l'algorithme EGO au dimensionnement optimal d'un circuit analogique. Nous comparons sa rapidité et sa précision par rapport aux métaheuristiques basées sur la simulation.

Le troisième chapitre détaille les améliorations que nous avons proposées à cet algorithme pour l'évaluation de problèmes mono- et multiobjectif. Dans sa première partie, nous comparons la précision de l'algorithme EGO par rapport aux autres métaheuristiques basées sur la technique de métamodélisation "*krigeage*" en utilisant la métrique « *Wilcoxon Signed Rank Test* ». Dans la deuxième partie, nous effectuons une intégration du critère « *Pseudo expected improvement* » (PEI) dans l'algorithme EGO pour le dimensionnement de circuits analogiques. Dans la dernière partie, nous présentons une nouvelle approche pour développer l'algorithme EGO multiobjectif en se basant sur le critère « *Expected Improvement Matrix* » (EIM) pour réduire le temps de calcul des critères (EI).

Finalement, une conclusion clôturera le mémoire et soulignera quelques perspectives à ce travail.

Chapitre 1 : Métaheuristiques et métamodélisation : état de l'art

1.1. Introduction

La conception de circuits analogiques, leurs structures, leurs organisations et leurs polarisations doivent être optimisées. La performance des circuits analogiques peut être optimisée en se basant soit sur les équations, soit sur les simulations. L'optimisation basée sur les équations est rapide, mais relativement peu précise. Par contre, l'optimisation basée sur les simulations est précise, mais sa durée d'exécution est très importante. Afin de tirer profit des avantages de chaque technique, nous avons opté pour la métamodélisation.

Dans ce chapitre, nous présentons un état de l'art des méthodes d'optimisation mono-objectif et multiobjectif et nous mettons en exergue l'intérêt du recours à ces métaheuristiques. Ensuite, nous décrivons les deux processus d'évaluation de dimensionnement des circuits analogiques. Le premier processus est basé sur les équations et le deuxième processus est basé sur la simulation. Enfin, nous présentons les modèles de substitution et les différentes techniques de métamodélisation.

1.2. Les techniques d'optimisation

La résolution des problèmes d'optimisation est devenue un sujet central en recherche opérationnelle, le nombre des problèmes d'aide à la décision pouvant être formalisés sous la forme d'un problème d'optimisation étant en forte croissance. Ces problèmes ont pour objectif de minimiser (ou maximiser) une ou plusieurs fonction(s) objectif(s) tout en respectant certains critères. Les problèmes d'optimisation peuvent être classés en problèmes d'optimisation mono-objectif (l'intérêt est d'optimiser une seule fonction) et problèmes d'optimisation multiobjectif (l'intérêt est d'optimiser deux ou plusieurs fonctions). Différentes techniques sont utilisées pour résoudre des problèmes d'optimisation complexes, parmi ces techniques, nous pouvons citer les métaheuristiques qui sont apparues au début des années 1970 [Holland, 1975]. Les métaheuristiques sont des techniques d'optimisation approchées qui produisent généralement des solutions très proches de l'optimum en un temps de calcul réduit.

Nous nous intéressons dans ce qui suit à présenter les techniques d'optimisation mono-objectif et multiobjectif. Un intérêt particulier est accordé aux quatre métaheuristiques mono-

objectif : les algorithmes génétiques (GA), l'algorithme à évolution différentielle (DE), l'algorithme 'Backtracking Search Algorithm' (BSA) et l'algorithme d'optimisation par essaim particulaire (PSO). Aussi, deux autres métaheuristiques multiobjectif sont présentées : l'algorithme 'Non Dominated Sorting Genetic Algorithm-II' (NSGA-II) et l'algorithme d'optimisation par essais de particules dans sa version multiobjectif (MOPSO).

1.2.1. L'optimisation mono-objectif

1.2.1.1. Définition du problème

Un problème d'optimisation mono-objectif se définit comme la recherche de la solution optimale (minimum ou maximum) d'une seule fonction objectif. Généralement, un problème d'optimisation peut présenter des contraintes d'égalité et/ou d'inégalité sur les solutions candidates.

D'un point de vue mathématique, un problème d'optimisation mono-objectif se présente de cette façon :

$$\left\{ \begin{array}{l} \text{Minimiser } f(\vec{x}) \\ \text{Tel que } g(\vec{x}) \leq 0 \\ \text{Avec } \vec{x} \in \mathbb{R}^n, g(\vec{x}) \in \mathbb{R}^q \end{array} \right. \quad (1.1)$$

où \vec{x} est le vecteur de décision et f la fonction à optimiser avec n paramètres, et g les q contraintes d'égalité et/ou d'inégalité à satisfaire.

Il existe de nombreuses méthodes déterministes ou "exactes" qui permettent de résoudre certains types de problèmes d'optimisation en un temps fini. Cependant, ces méthodes nécessitent que la fonction objectif présente un certain nombre de caractéristiques, telles que la convexité, la continuité ou encore la dérivabilité. Parmi les méthodes les plus connues, on peut citer les méthodes de programmation linéaire, la méthode quadratique ou dynamique ou aussi la méthode du gradient.

Certains problèmes restent cependant trop complexes à résoudre par les méthodes déterministes, vu que certaines caractéristiques présentent des problèmes (une relation exponentielle de la complexité en fonction du temps de calcul). Dans ce cas, le problème d'optimisation est dit difficile, car aucune méthode déterministe ne peut le résoudre en un temps raisonnable.

Ces problèmes d'optimisation difficiles se divisent en deux catégories [Siarry *et al.*, 1997]: les problèmes à variables discrètes et les problèmes à variables continues. D'une façon générale, un problème d'optimisation à variables discrètes ou combinatoires consiste à trouver, dans un ensemble discret, la meilleure solution réalisable et ceci au sens du problème défini par

l'équation (1.1). Le problème majeur réside ici dans le fait que le nombre de solutions réalisables est généralement très élevé, donc il est très difficile de trouver la meilleure solution en un temps "raisonnable". L'utilisation d'algorithmes d'optimisation stochastiques, tels que les métaheuristiques, permet de trouver une solution approchée en un temps relativement réduit [Dréo *et al.*, 2003] [Clerc *et al.*, 2004].

Par définition, les métaheuristiques sont des algorithmes adaptables à un grand nombre de problèmes différents sans changements majeurs dans l'algorithme [Siarry *et al.*, 2007]. On distingue deux catégories de métaheuristiques :

- Les métaheuristiques de voisinage :

Ce type d'algorithme fait évoluer une seule solution sur l'espace de recherche local à chaque itération puis la compare aux optimums (solutions trouvées au cours des générations). Les algorithmes d'optimisation les plus connus dans cette classe sont le recuit simulé [Kirkpatrick *et al.*, 1983] et la recherche avec tabou [Glover, 1989].

- Les métaheuristiques distribuées :

Elles sont appelées aussi métaheuristiques à population. Elles utilisent un échantillonnage de fonctions objectifs comme base d'apprentissage. Ces dernières deviennent alors extrêmement importantes lors du choix de l'espace de recherche local. Dans cette famille, les métaheuristiques utilisent la notion de population : elles manipulent un ensemble de solutions en parallèle. Chaque élément de la population parcourt un certain nombre de solutions dans l'ensemble local. Parmi les algorithmes inclus dans cette classification, on peut citer les algorithmes évolutionnaires et les algorithmes basés sur l'intelligence des essaims.

La figure 1.1 présente une classification des différents algorithmes d'optimisation mono-objectif [Dréo *et al.*, 2003].

1.2.1.2. Les algorithmes évolutionnaires

Dans ce qui suit, nous nous intéressons à présenter les trois algorithmes évolutionnaires les plus utilisés dans la littérature.

- **Les algorithmes génétiques :**

Les algorithmes génétiques (GA) sont des techniques évolutionnaires généralement utilisées dans les problèmes d'optimisation. Ils sont basés sur le processus d'évolution génétique des organismes biologiques à travers les générations selon la théorie de l'évolution de Darwin [Darwin *et al.*, 1859].

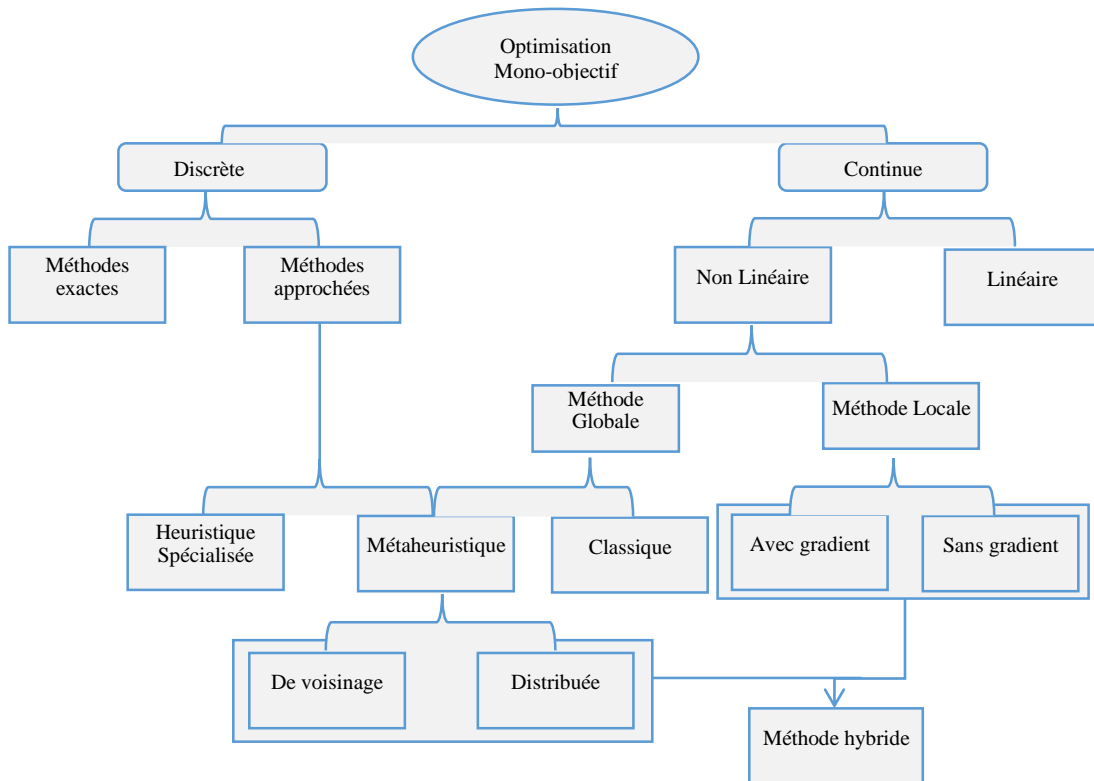


Figure 1.1 : Classification des méthodes d'optimisation

Les algorithmes évolutionnaires appliqués à un problème d'optimisation font évoluer un ensemble de solutions candidates, constituant une population d'individus (ou de chromosomes). Un individu représente une solution possible à chaque problème donné. Après une mesure de la qualité de chaque individu par rapport à sa fonction d'adaptation, une nouvelle population est produite en sélectionnant les parents parmi les meilleurs individus de la génération actuelle pour effectuer des croisements et des mutations.

La nouvelle population contient ainsi une plus grande proportion des caractéristiques des meilleurs individus de la génération précédente. De cette façon, les meilleurs gènes se propagent dans la population en se combinant ou en échangeant les meilleurs traits.

La population initiale peut être obtenue par génération aléatoire, de façon heuristique ou directement choisie par l'utilisateur.

La figure 1.2 présente l'organigramme des algorithmes génétiques.

- La sélection a pour objectif d'identifier les individus qui doivent se reproduire. Elle doit favoriser les meilleurs éléments selon un critère à optimiser.
- Le croisement permet de conserver les meilleures caractéristiques des parents. Il consiste à échanger une partie du matériel génétique des parents pour former deux nouveaux individus. Ces individus s'appellent enfants possédant des caractéristiques issues de deux parents.

➤ La mutation est définie comme étant l'inversion d'un bit dans un chromosome (le cas du codage binaire). Cela revient à modifier aléatoirement la valeur d'un paramètre du dispositif. Elle joue le rôle de bruit et empêche l'évolution de se figer. Elle permet d'assurer une recherche aussi bien globale que locale, selon le poids et le nombre des bits mutés. De plus, elle permet de ne pas être piégé dans un optimum local.

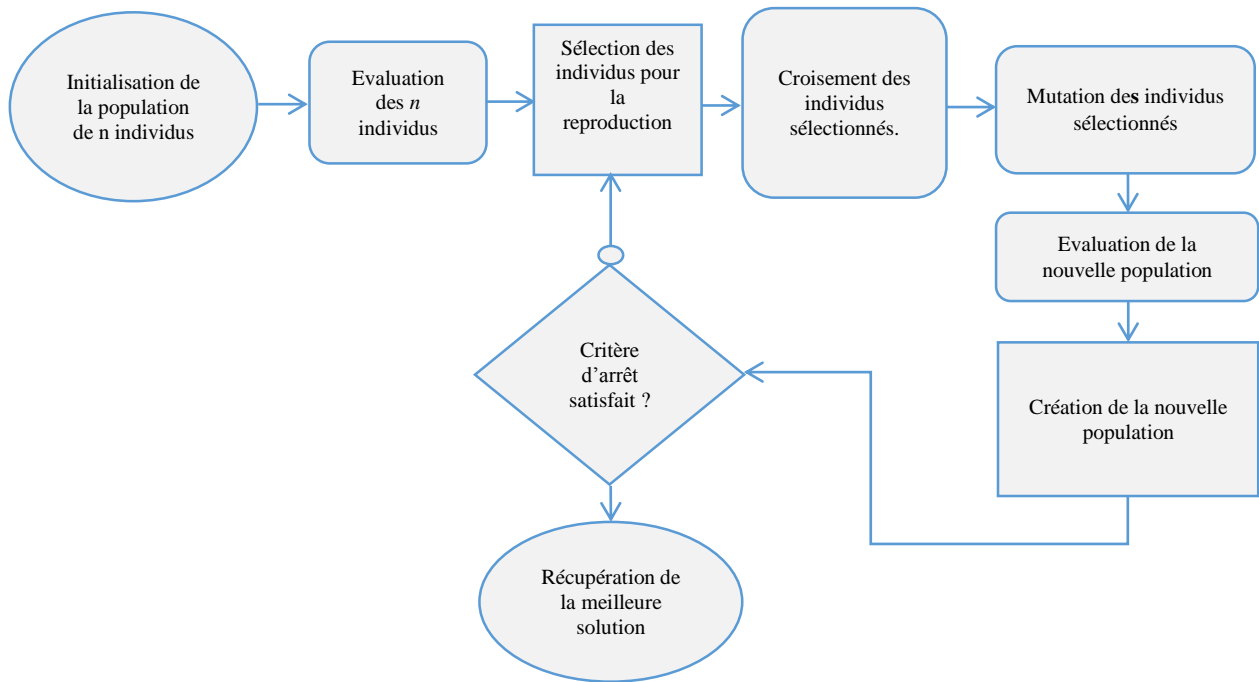


Figure 1.2 : Organigramme des algorithmes génétiques [Mancer, 2012]

▪ L'algorithme de l'évolution différentielle :

L'algorithme à évolution différentielle (DE) '*Differential Evolution*' a été proposé par R. Storn et K. Price dans les années 1990 [Storn *et al.*, 1997]. Cet algorithme est une métaheuristique stochastique qui a été inspirée par les algorithmes génétiques et les stratégies évolutionnaires combinées avec une technique géométrique de recherche. Les algorithmes génétiques changent la structure des individus en utilisant la mutation et le croisement, alors que les stratégies évolutionnaires réalisent l'auto-adaptation par une manipulation géométrique des individus [Bres *et al.*, 2006] [Guerra, 2016] [Bres *et al.*, 2007].

Dans cet algorithme, la population initiale est générée par tirage aléatoire uniforme sur l'ensemble des valeurs possibles de chaque variable. Les bornes inférieures et supérieures des variables sont spécifiées par le concepteur selon la nature du problème. Après l'initialisation, l'algorithme effectue une série de transformations sur les individus, dans un processus appelé évolution. La population contient N individus. Chaque individu $x_{i,G}$ est un vecteur de dimension D , où G désigne la génération [El Dor, 2012] :

$$x_{i,G} = (x_{1i,G}, x_{2i,G}, \dots, x_{Di,G}) \text{ avec } i = 1, 2, \dots, N \quad (1.2)$$

L'algorithme DE utilise les mêmes techniques (mutation, croisement et sélection) des algorithmes génétiques. A chaque génération, l'algorithme applique successivement ces trois opérations sur chaque vecteur pour produire un vecteur d'essai (*trial vector*) :

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \dots, u_{Di,G+1}) \text{ avec } i = 1, 2, \dots, N \quad (1.3)$$

Une opération de sélection permet de choisir les individus à conserver pour la nouvelle génération (G + 1).

➤ **Mutation**

Pour chaque vecteur courant $x_{i,G}$, on génère un vecteur mutant $v_{i,G+1}$ qui peut être créé en utilisant une des stratégies de mutation suivantes :

✓ **Rand/1 :**

$$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) \quad (1.4)$$

✓ **Best/1 :**

$$v_{i,G+1} = x_{best,G} + F \cdot (x_{r1,G} - x_{r2,G}) \quad (1.5)$$

✓ **Current to best/1 :**

$$v_{i,G+1} = x_{i,G} + F \cdot (x_{r1,G} - x_{r2,G}) + F \cdot (x_{best,G} - x_{i,G}) \quad (1.6)$$

✓ **Best/2 :**

$$v_{i,G+1} = x_{best,G} + F \cdot (x_{r1,G} - x_{r2,G}) + F \cdot (x_{r3,G} - x_{r4,G}) \quad (1.7)$$

✓ **Rand/2 :**

$$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) + F \cdot (x_{r4,G} - x_{r5,G}) \quad (1.8)$$

Les indices r_1, r_2, r_3, r_4 et $r_5 \in \{1, 2, \dots, N\}$ sont des entiers aléatoires et ils sont tous différents. Ils sont également choisis différents de l'indice courant i . $x_{best,G}$ est le meilleur individu à la $G^{\text{ème}}$ génération. $F \in [0, 2]$ est une valeur constante, appelée *differential weight*, qui contrôle l'amplification de la variation différentielle de $(x_{ri,G} - x_{rj,G})$.

➤ **Croisement**

Après la mutation, une opération de croisement binaire forme le vecteur d'essai final $u_{i,G+1}$, selon le vecteur $x_{i,G}$ et le vecteur mutant correspondant $v_{i,G+1}$. L'opération de croisement est introduite pour augmenter la diversité des vecteurs de paramètres perturbés.

Le nouveau vecteur $u_{i,G+1}$ est donné par la formule suivante :

$$u_{i,G+1} = \begin{cases} u_{1i,G+1} & \text{si } (\text{randb}(j) \leq CR) \text{ ou } j = \text{rnbr}(i) \\ u_{ji,G} & \text{si } (\text{randb}(j) > CR) \text{ ou } j = \text{rnbr}(i) \end{cases} \quad (1.9)$$

Pour tout $j \in \{1, 2, \dots, D\}$

où $randb(j)$ est la $j^{\text{ème}}$ valeur procurée par un générateur de nombres aléatoires uniformes appartenant à l'intervalle $[0,1]$. CR est le coefficient de croisement qui appartient à l'intervalle $[0,1]$ et est déterminé par l'utilisateur.

$rnbr(i)$ est un indice choisi au hasard dans l'ensemble $\{1, 2, \dots, N\}$.

➤ **Sélection**

Pour décider quel vecteur, parmi $u_{i,G+1}$, ou $x_{i,G}$, doit être choisi dans la génération $G + 1$, on doit comparer les valeurs de fonction du coût de ces deux vecteurs. En effet, on garde le vecteur ayant la plus petite valeur de fonction du coût en cas de minimisation. Le nouveau vecteur $x_{i,G+1}$ est choisi selon l'expression suivante:

$$x_{i,G+1} = \begin{cases} u_{1i,G+1} & \text{si } f(u_{i,G+1}) < f(x_{i,G}) \\ x_{i,G} & \text{sinon} \end{cases} \quad (1.10)$$

Un bon réglage des principaux paramètres de l'algorithme (taille de la population N , facteur de mutation F et facteur de croisement CR) contribue de façon importante à l'efficacité de la méthode [El Dor, 2012].

▪ **BSA (Backtracking Search Optimization Algorithm)**

L'algorithme BSA (*Backtracking Search Algorithm*) est un algorithme évolutionnaire, développé pour résoudre les problèmes d'optimisation continue [Brévilliers *et al.*, 2015]. Une des particularités de cette approche se présente dans l'utilisation d'une mémoire, pour conserver une population précédente, qui intervient dans l'opérateur de mutation mis en œuvre. Il est par ailleurs conçu autour d'un unique paramètre de contrôle qui est utilisé au cours du processus de croisement. La structure de BSA est simple, et les opérateurs de mutation et de croisement utilisés lui permettent de s'adapter à des problèmes variés et de les résoudre efficacement [Civicioglu, 2013] [Brévilliers *et al.*, 2015].

La figure 1.3 présente l'organigramme de l'algorithme BSA.

A chaque génération, l'algorithme applique successivement ces cinq opérations : initialisation, sélection 1, mutation, croisement et sélection 2. Les opérateurs de mutation et de croisement utilisés permettent à l'algorithme BSA de s'adapter à des problèmes variés et de les résoudre efficacement.

Dans ce qui suit, nous présentons l'algorithme qui se base sur l'intelligence des essais.

1.2.1.3. L'optimisation par essais particuliers

La technique d'optimisation par essais particuliers (PSO) est basée sur l'intelligence des essais, où la population s'appelle un essaim et chaque individu s'appelle une particule.

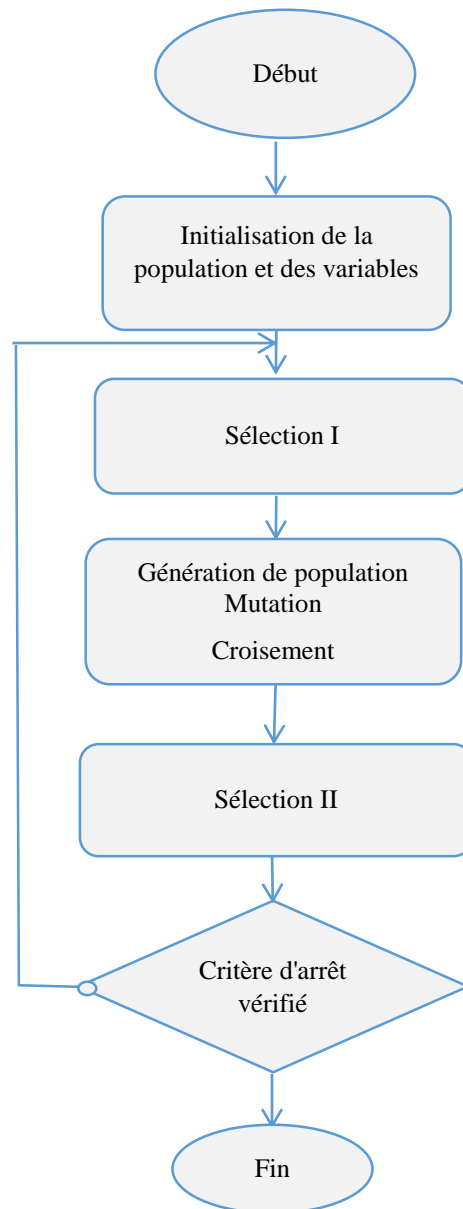


Figure 1.3 : Organigramme de l'algorithme BSA

L'optimisation par essaim particulaire est une méthode stochastique proposée en 1995 sous le nom de '*Particle Swarm Optimization*' (PSO) [Kennedy *et al.*, 1995]. Pour appliquer l'algorithme PSO, il faut définir un espace de recherche constitué de particules et une fonction objectif à optimiser. Le principe de l'algorithme est de déplacer ces particules afin qu'elles trouvent l'optimum. Une particule i est caractérisée par une position \vec{X}_i et une vitesse de changement de position \vec{V}_i . Elle garde en mémoire sa meilleure position atteinte. Pour chaque itération t :

- Chaque particule est capable d'évaluer la qualité de sa position et de garder en mémoire sa meilleure performance.

- Chaque particule est capable de communiquer avec un certain nombre de ses congénères, qui s'appellent aussi informatrices, et d'obtenir pour chacune d'elles une valeur de performance.

À chaque pas de temps, chaque particule connaît :

- Sa meilleure position visitée. On retient essentiellement la valeur du critère calculé ainsi que ses coordonnées.
- La position du meilleur voisin de l'essaim qui correspond à l'ordre optimal.
- La valeur qu'elle donne à la fonction objectif car à chaque itération, il faut comparer entre la valeur du critère donnée par la particule courante et la valeur optimale.

La figure 1.4 illustre comment une particule se déplace dans un essaim :

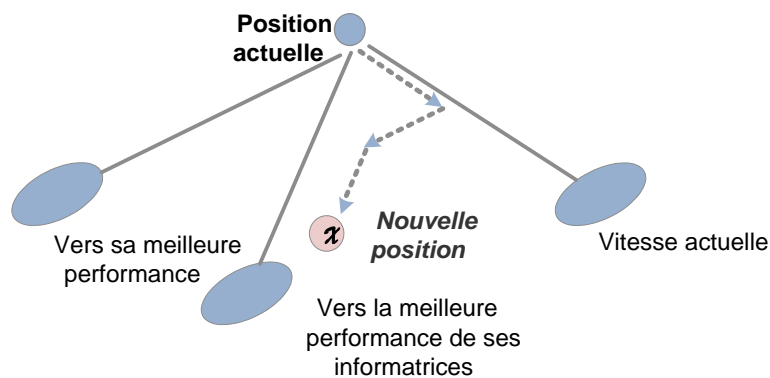


Figure 1.4 : Déplacement d'une particule

A chaque itération, les particules sont mises à jour en tenant compte de la meilleure position \vec{P}_i de la particule et de la meilleure position \vec{P}_g de ses voisins suivant la formule citée ci-dessous :

$$\vec{V}_i(t) = \underbrace{\omega(t-1) \cdot \vec{V}_i(t-1)}_{\text{Inertie}} + \underbrace{c_1 r_1 (\vec{P}_i - \vec{X}_i(t-1))}_{\text{Influence personnelle}} + \underbrace{c_2 r_2 (\vec{P}_g - \vec{X}_i(t-1))}_{\text{Influence sociale}} \quad (1.11)$$

$$\vec{X}_i(t) = \vec{X}_i(t-1) + \vec{V}_i(t) \quad (1.12)$$

Pour l'optimisation par essais particulaires, le facteur d'inertie permet de contrôler la phase de la diversification. r_1 et r_2 sont des variables aléatoires comprises dans l'intervalle $[0,1]$, c_1 et c_2 sont les facteurs d'apprentissage appelés aussi coefficients d'accélération. c_1 représente l'attraction de la particule vers la meilleure position, c_1 est la mémoire propre de particule. c_2 représente l'attraction vers la meilleure position de ses voisins. Ces deux constantes positives sont déterminées de façon empirique et suivant la relation $c_1 + c_2 \leq 4$ [Mancer, 2012] et ω est le facteur de pondération appelé aussi coefficient d'inertie. Il joue un rôle important dans la

convergence, il permet de définir la capacité d'exploration de chaque particule en vue d'améliorer la convergence de la méthode. Une faible valeur de ω (< 1) permet une recherche locale (exploration locale). Tandis qu'une grande valeur de ω (> 1) permet d'explorer globalement l'espace de recherche. Fixer ce facteur, revient donc à trouver un compromis entre l'exploration locale et l'exploration globale.

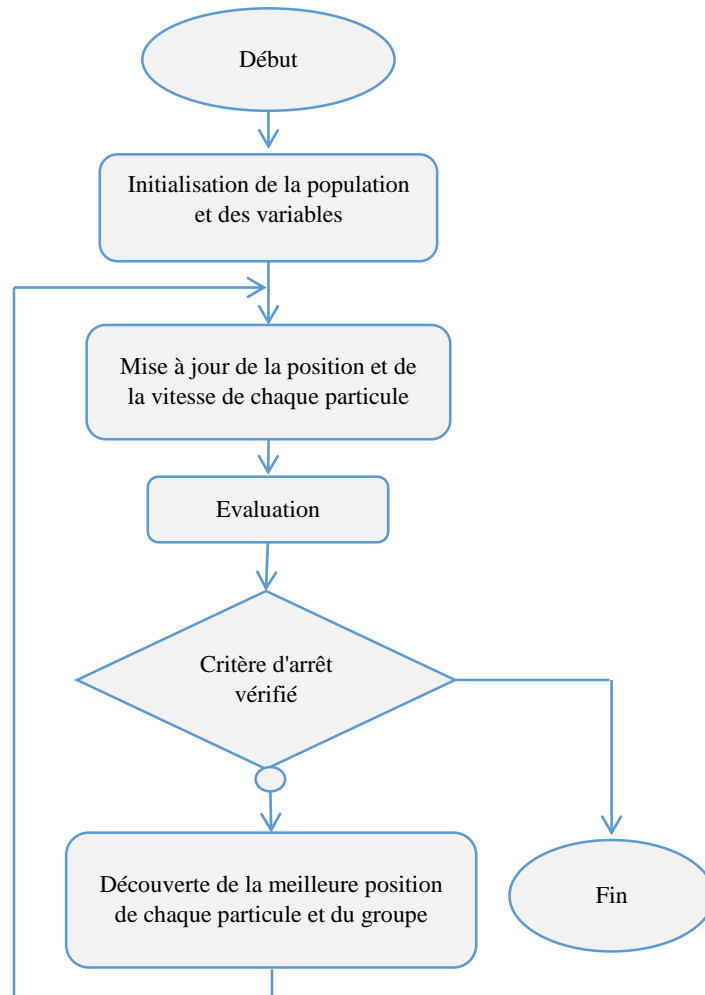


Figure 1.5 : Schéma de principe de l'algorithme PSO [Mancer, 2012]

Dans ce qui suit, nous détaillons l'optimisation multiobjectif et deux algorithmes métaheuristiques les plus utilisés dans la littérature.

1.2.1. L'optimisation multiobjectif

1.2.2.1. Définition du problème

Le but principal de l'optimisation mono-objectif est de trouver la solution optimale globale qui produit une meilleure valeur de la fonction mono-objectif. Dans un problème multiobjectif, il y a plusieurs fonctions objectifs qui doivent être optimisées simultanément. Un problème

multiobjectif a pour but de trouver un ensemble de points appelé l'ensemble des meilleurs compromis ou le front de Pareto [Pareto, 1964] [Coello *et al.*, 2002].

D'une façon générale, les problèmes d'optimisation multiobjectif sont définis, comme suit :

$$\left| \begin{array}{ll}
 \text{Minimiser } \vec{f}(\vec{x}) = \vec{f}_1(\vec{x}), \vec{f}_2(\vec{x}), \dots, \vec{f}_m(\vec{x}) & (m \text{ fonctions à optimiser}) \\
 \text{sous les contraintes : } \vec{g}(\vec{x}) \leq 0 & (q \text{ inégalités à satisfaire}) \\
 \vec{h}(\vec{x}) = 0 & (p \text{ égalités à satisfaire}) \\
 \text{avec } \vec{x} \in \mathbb{R}^n, \vec{f}(\vec{x}) \in \mathbb{R}^m, \vec{g}(\vec{x}) \in \mathbb{R}^q \text{ et } \vec{h}(\vec{x}) \in \mathbb{R}^p &
 \end{array} \right. \quad (1.13)$$

où \vec{x} est le vecteur de décision, \vec{f} : fonction à optimiser, n: paramètres, m : fonctions à optimiser, (q+p) : contraintes à satisfaire.

1.2.2.2. Relations d'ordre et de dominance

Le but principal de l'optimisation multiobjectif est de fournir un meilleur compromis entre les différents objectifs sous forme de surface de compromis appelée *front de Pareto*. La notion d'optimum change par Front de Pareto [Deb, 2001] [Coello *et al.*, 2002]. Les solutions "Pareto optimales" sont connues sous le nom de solutions "non-dominées".

Dans le cas multiobjectif, pour mettre en évidence la relation de dominance entre la solution considérée et les autres solutions, nous introduisons les définitions suivantes :

Définition 1 : Etant donné $\vec{x}, \vec{y} \in \mathbb{R}^n$, on dit que $\vec{x} \leq \vec{y}$ si $x_i \leq y_i$ pour $i = 1, \dots, k$, et que \vec{x} domine \vec{y} ($\vec{x} < \vec{y}$) si $\vec{x} \leq \vec{y}$ et $\vec{x} \neq \vec{y}$.

Définition 2 : On dit qu'un vecteur solution $\vec{x} \in X \subset \mathbb{R}^D$ est non-dominé dans X s'il n'existe pas d'autre vecteur $\vec{x}' \in X$ tel que $\vec{f}(\vec{x}') < \vec{f}(\vec{x})$.

Définition 3 : On dit qu'un vecteur de solutions $\vec{x}^* \in F \subset \mathbb{R}^D$ est Pareto-optimal s'il est non dominé dans F .

Les solutions qui dominent les autres, mais ne se dominent pas entre elles sont appelées solutions optimales au sens de Pareto. La figure 1.6 illustre la notion de dominance et les solutions Pareto pour un problème de minimisation à deux objectifs.

1.2.2.3. Frontière de Pareto

La frontière ou Front de Pareto est l'ensemble de tous les points Pareto-optimaux. Elle utilise directement la notion de dominance dans la sélection des solutions générées, contrairement aux autres approches qui utilisent une fonction d'utilité ou qui traitent séparément les différents objectifs.

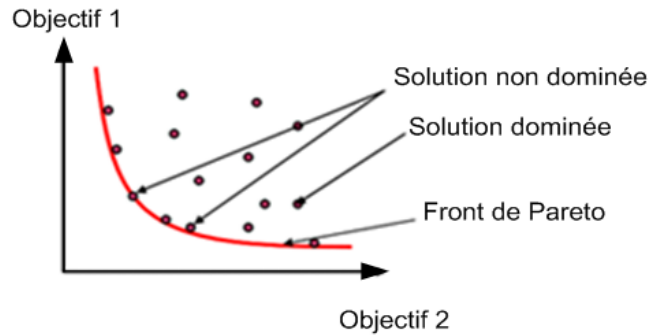


Figure 1.6 : *Front de Pareto pour un problème bi-objectif*

La figure 1.7 représente pour un problème convexe à deux objectifs les quatre frontières de Pareto en fonction du désir de l'utilisateur pour minimiser ou maximiser les objectifs:

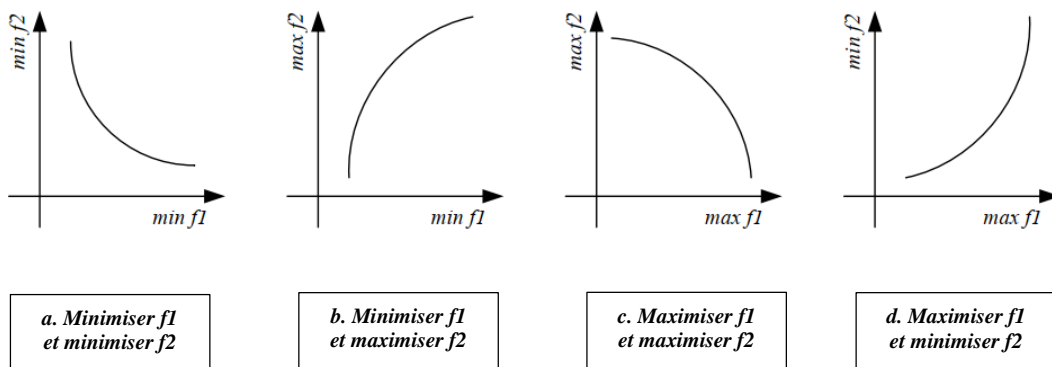


Figure 1.7 : *Formes des fronts de Pareto*

Ces méthodes se sont avérées être les plus efficaces, la majorité des algorithmes d'optimisation utilisent cette approche de Pareto [Deb, 2001] [Coello *et al.*, 2002]. Ci-après nous présentons brièvement les principales métaheuristiques traitées dans la littérature et utilisées pour l'optimisation des circuits analogiques.

1.2.2.4. Métaheuristiques pour l'optimisation multiobjectif

Les métaheuristiques ont toutes été conçues à l'origine pour résoudre des problèmes mono-objectif. Cependant, devant l'omniprésence des problèmes multiobjectif dans les cas réels, de nouvelles méthodes ont dû être adaptées pour résoudre de tels problèmes.

Les métaheuristiques sont des méthodes visant à résoudre des problèmes d'optimisation difficiles. La plupart des métaheuristiques multiobjectif sont des algorithmes évolutionnaires qui sont très connus dans la littérature [Engrand, 1998] [Deb, 2001] [Coello *et al.*, 2002] [Siarry *et al.*, 2007].

Dans ce qui suit, nous présentons les principales métaheuristiques multiobjectif. Ces métaheuristiques seront par la suite utilisées pour l'optimisation des performances de circuits analogiques.

▪ Les Algorithmes Génétiques multiobjectif

Un nombre important d'algorithmes génétiques multiobjectif a été proposé dans la littérature dont on mentionne : VEGA '*Vector Evaluated Genetic Algorithm*' [Deb, 2001], NPGA '*Niched Pareto Genetic Algorithm*' [Horn *et al.*, 1994], NPGA 2 [Zitzler, 1999], NSGA '*Non Dominated Sorting Genetic Algorithm*' [Srinivas *et al.*, 1994], NSGA-II '*Non Dominated Sorting Genetic Algorithm-II*' [Deb *et al.*, 2002].

Dans ce qui suit, nous détaillons l'algorithme le plus utilisé dans la littérature, à savoir le NSGA-II qui est connu comme un algorithme de référence pour l'évaluation d'autres algorithmes multiobjectif. Les opérateurs génétiques de croisement et de mutation peuvent affecter le meilleur individu d'une génération. Le modèle élitiste a pour avantage d'écartier la possibilité de perdre cet individu. Ce modèle copie le meilleur individu de chaque génération dans la population de la génération suivante. Ce modèle peut accélérer la vitesse de domination exercée par cet individu sur la population.

Dans cet algorithme, une population de parents (P_t) de taille (N) et une population d'enfants (Q_t) de taille (N) sont assemblées pour former une population ($R_t = P_t \cup Q_t$). Comme le montre la figure 1.8, cet assemblage permet d'assurer l'élitisme. Cette population de taille ($2N$) est ensuite triée selon un critère de non-dominance pour identifier et pour classer les différents fronts F_1, F_2 , etc. Les meilleurs individus vont se retrouver dans le (ou les) premier(s) front(s). Une nouvelle population parent (P_{t+1}) est formée (premier front F_1 , second front F_2 , etc.) tant que le nombre d'individus contenus dans ceux-ci ne dépasse pas N . Si le nombre d'individus présents dans (P_{t+1}) est inférieur à (N), une procédure de *crowding* est appliquée sur le premier front suivant, (F_i), non inclus dans (P_{t+1}). Le but de cet opérateur est d'insérer les ($N - |P_{t+1}|$) meilleurs individus qui manquent dans la population (P_{t+1}). Les individus de ce front sont utilisés pour calculer la distance de *crowding* entre deux solutions [Deb *et al.*, 2002].

Une fois que les individus appartenant à la population (P_{t+1}) sont identifiés, une nouvelle population enfant (Q_{t+1}) est créée par sélection, croisement et mutation. La sélection par tournoi [Goldberg, 1989] est utilisée mais le critère de sélection est basé sur l'opérateur de comparaison. Le processus est itéré, d'une génération à la suivante, jusqu'à ce que le critère d'arrêt soit satisfait.

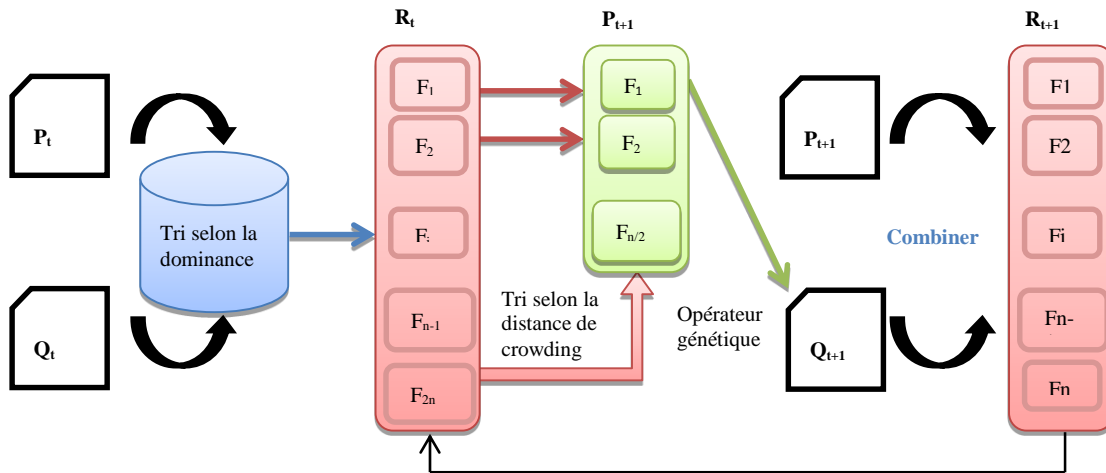


Figure 1.8 : Principe de l'algorithme NSGA II

▪ L'optimisation multiobjectif par essaim particulaire

Il existe différents algorithmes multiobjectif basés sur l'optimisation par essaim particulaire. On distingue le MOPSO 'Multi-objective Particle Swarm Optimization' [Coello et al., 2002] le MOPSO-CD 'Multi-Objective Particle Swarm Optimization using the Crowding Distance technique' utilisant la technique 'Crowding Distance' [Raquel et al., 2005].

Nous nous intéressons particulièrement dans ce qui suit à l'algorithme MOPSO-CD car cet algorithme incorpore le mécanisme de calcul de la distance de *crowding* et un opérateur de mutation qui sert à maintenir la diversité des solutions non dominées dans l'archive extérieure.

Après initialisation de l'essaim, un ensemble de 'leaders' est également initialisé avec les particules non-dominées de l'essaim (l'ensemble de 'leaders' est souvent stocké dans des archives externes). Ensuite, une mesure de qualité est calculée pour tous ces 'leaders' afin de choisir, un 'leader' pour chaque particule de l'essaim.

A chaque génération et pour chaque particule, un 'leader' est choisi et il identifie sa position. La plupart des algorithmes multiobjectif classiques utilisant le PSO appliquent un opérateur de mutation après l'exécution de la position. La particule est ensuite évaluée et la valeur de P_{best} (la meilleure position que la particule a atteinte jusqu'ici) correspondante est mise à jour. P_{best} est mis à jour quand la particule atteint une meilleure position. Après la mise à jour de toutes les particules, l'ensemble de 'leaders' est mis à jour aussi. Finalement, la mesure de qualité de l'ensemble est recalculée. Ce processus est répété jusqu'à l'atteinte du nombre maximum des itérations [Kotti, 2017].

Il est important de préciser que le mécanisme de calcul de la distance de *crowding* est nécessaire pour :

- Le choix global des solutions non dominées.

- Le maintien de la diversité des solutions.

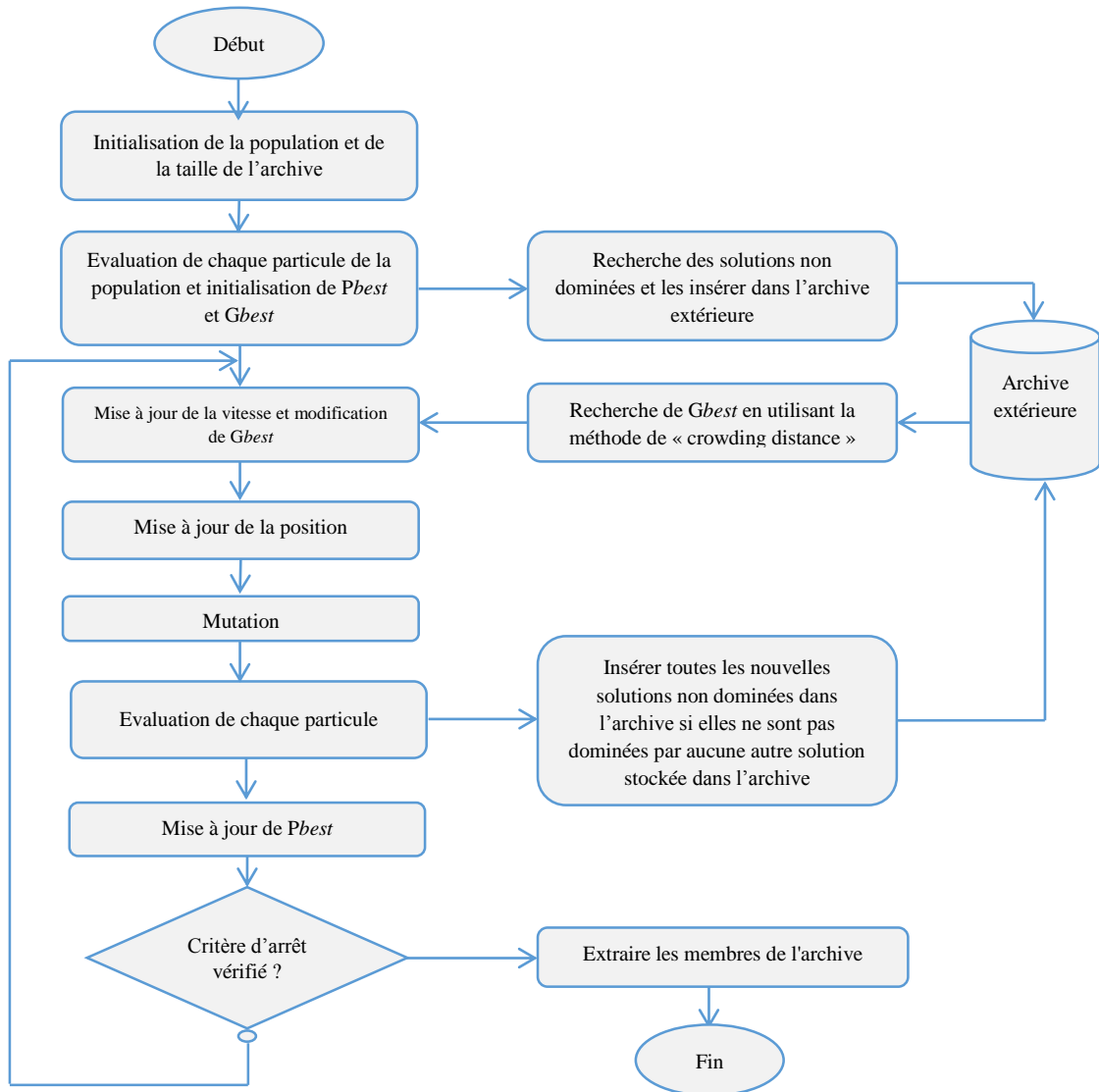


Figure 1.9 : L'organigramme du MOPSO-CD

Dans ce qui suit, nous présentons les principaux outils de dimensionnement des circuits analogiques.

1.3. Application des métaheuristiques aux problèmes de dimensionnement des circuits analogiques

L'augmentation de la complexité de la tâche de conception de circuits analogiques, l'effet des variations des technologies d'intégration et la demande de réduction du temps de conception, ont augmenté la nécessité de développer de nouveaux outils d'automatisation de la conception et de dimensionnement des circuits analogiques [Medeiro *et al.*, 1994] [Dastidar *et al.*, 2005]. Des récents progrès dans l'automatisation de la conception des circuits analogiques

ont amené à une transition progressive de la conception basée sur l'expérience et les connaissances du concepteur [Degrauwe *et al.*, 1987] [Harjani *et al.*, 1989] [El-Turky *et al.*, 1989] [Conn *et al.*, 1996] vers des méthodologies de conception basées sur des algorithmes d'optimisation.

Dans ce qui suit, nous présentons les principaux outils de dimensionnement des circuits analogiques en particulier ceux qui sont basés sur l'optimisation [Barros *et al.*, 2010]. Il existe deux processus d'évaluation : le premier processus est basé sur les équations et le deuxième est basé sur la simulation (connue sous le nom de '*inloop technique*'). Ces deux processus seront détaillés et mis en relief dans la suite de ce chapitre.

1.3.1. La technique d'optimisation basée sur les équations

Il s'agit d'extraire un jeu d'équations symboliques en utilisant le schéma interne du circuit et les modèles simplifiés des composants. Les équations peuvent être obtenues manuellement ou en utilisant des analyseurs symboliques tels que : OPASYN [Koh *et al.*, 1990], SAPWIN [Huelsman, 1996], CASCADES [Fakhfakh *et al.*, 2010]. L'avantage de cette technique est principalement dans le temps d'exécution réduit puisque l'évaluation des performances est effectuée par un calcul direct d'équations symboliques [Fernandez *et al.*, 1996]. Néanmoins, l'inconvénient de l'utilisation des équations simplifiées est qu'elle fait accroître l'efficacité du calcul aux dépens de la précision [Sallem, 2015].

1.3.2. La technique d'optimisation basée sur la simulation

La méthodologie de conception basée sur l'optimisation est en relation directe avec un simulateur tel que SPICE '*Simulation Program with Integrated Circuit Emphasis*' [Sallem *et al.*, 2009] [Barros *et al.*, 2010]. Ces simulateurs sont utilisés pour évaluer les performances et les contraintes du circuit. En effet, il s'agit de caractériser le circuit par une série de simulations et d'en extraire ainsi un ensemble de données reliant les performances du circuit à un vecteur de variables d'entrée. Une série de simulations itératives est nécessaire pour aboutir à la satisfaction des spécifications.

Cette méthode a pour avantage d'avoir des résultats très précis étant donné que ce sont les modèles des composants du simulateur qui sont directement utilisés. Mais, l'inconvénient de cette technique est la durée d'exécution [Sawal Hamid, 2009]. Toutefois, ce facteur peut être atténué avec la progression continue des performances des outils informatiques. Cependant, l'augmentation incessante de la complexité des circuits et le besoin

gourmand en performances, font que ces temps de simulation restent une limitation majeure de cette approche.

Nous présentons dans ce qui suit la technique d'optimisation basée sur la simulation. L'idée de base consiste à "shunter" la phase de modélisation dans l'approche de dimensionnement. Pour cela, un simulateur est utilisé pour évaluer la fonction objectif tout en vérifiant les contraintes imposées. Dans ce processus, une métaheuristique est utilisée afin de chercher les valeurs optimales des dimensions des transistors du circuit. Un schéma de principe de cette approche est présenté dans la figure 1.10 [Sallem *et al.*, 2010] :

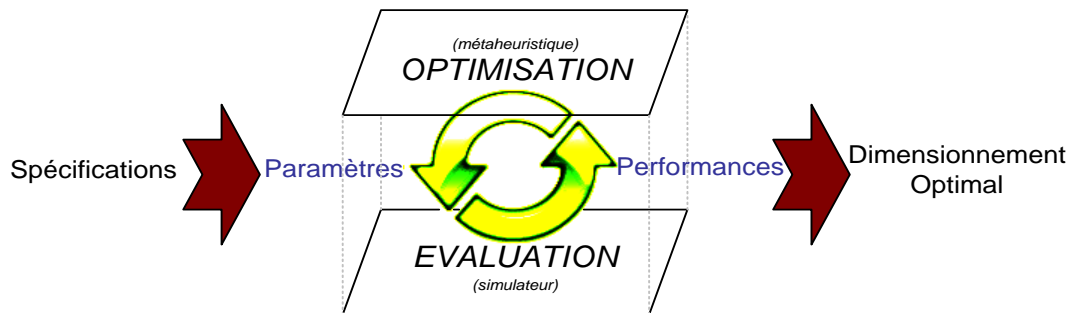


Figure 1.10 : *Technique de dimensionnement/d'optimisation basée sur la simulation.*

La figure 1.11 illustre l'organigramme de dimensionnement/optimisation des circuits analogiques basé sur la simulation [Sallem, 2015].

Le schéma de principe des différentes phases d'évaluation de cette métaheuristique (interface C++/Hspice) est présenté dans la figure 1.11 et il est détaillé comme suit :

- 1- Initialisation des paramètres du circuit,
- 2- Écriture du Netlist du circuit,
- 3- La métaheuristique fait appel au fichier Netlist, ouvre puis modifie les valeurs des paramètres du circuit (par exemple les dimensions des différents transistors MOS du circuit),
- 4- Le programme C++ (métaheuristique) fait appel au simulateur et lance la simulation du circuit sur la base du Netlist mis à jour,
- 5- La métaheuristique fait appel au fichier de sortie, généré suite à la simulation du Netlist du circuit, à partir duquel elle vérifie les contraintes du circuit,
- 6- Une fois que les contraintes sont vérifiées, le programme extrait du fichier de sortie les valeurs de/des fonction(s) objectif(s),
- 7- Les meilleures performances de la (les) fonction(s) objectif(s) seront archivées,

- 8- L'archive sera mise à jour à chaque itération jusqu'à la satisfaction du critère d'arrêt du programme.

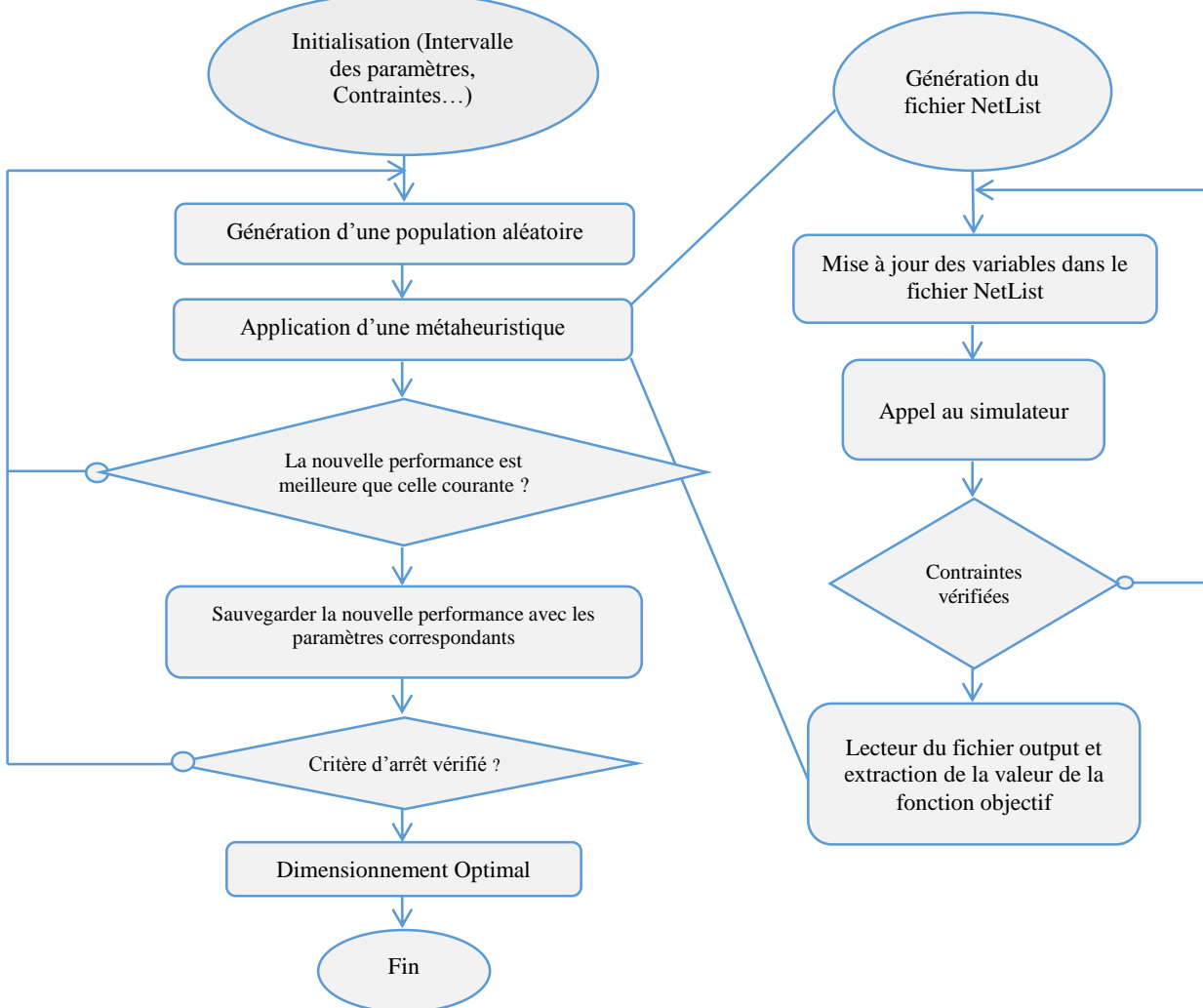


Figure 1. 11 : Organigramme de dimensionnement/optimisation des circuits analogiques basé sur la simulation [Sallem, 2015].

Bien que l'optimisation basée sur les simulations soit précise, elle présente un inconvénient à savoir un processus difficile et très couteux en temps de calcul (plusieurs jours). Un intérêt particulier est porté sur une nouvelle technique nommée la « métamodélisation », récemment proposée dans la littérature. Cette technique donne une bonne approximation des performances dans l'évaluation et elle se caractérise par sa rapidité.

1.4. Les modèles de substitution

La technique d'optimisation basée sur les simulations est bien précise, mais la durée d'exécution est très importante. En conséquence, nous optons pour une nouvelle technique de

dimensionnement/optimisation nommée la « métamodélisation », appelée également le modèle de substitution pour générer des modèles symboliques et précis.

Dans ce qui suit, nous définissons la nouvelle technique de métamodélisation et nous présentons les différentes techniques de génération des plans d'expériences et les techniques les mieux connues pour générer un modèle de substitution.

1.4.1. Définition

La métamodélisation consiste en une représentation mathématique du modèle de substitution construit à partir des points de base évalués par le modèle de simulation ou simplement des données expérimentales [Simpson *et al.*, 1998]. Les modèles de substitution permettent de remplacer un modèle lourd dans un processus d'optimisation afin de réduire le temps d'évaluation. Ils sont utilisés dans plusieurs domaines [Simpson *et al.*, 1998] [Jin *et al.*, 2001] [Forrester *et al.*, 2008].

Le défi de la modélisation est la génération d'un modèle précis en utilisant un nombre limité d'évaluations (simulations/mesures). Pour relever ce défi, le processus de modélisation comprend trois étapes principales qui se résument comme suit [Kotti, 2017] :

- La sélection des échantillons (la construction d'un plan d'expérience),
- La génération du modèle de substitution (le choix de la technique de modélisation),
- L'évaluation de la précision du modèle de substitution (la validation du modèle).

1.4.2. Les plans d'expériences

Initialement, un modèle de substitution est construit avec un nombre limité de points de données initiales choisis en utilisant les plans d'expériences. Ces points servent par la suite comme une entrée [Sack *et al.*, 1989].

Les buts attendus d'un plan d'expérience sont essentiellement l'échantillonnage de l'espace de recherche avec la bonne répartition des points (les points dans l'espace doivent être répartis le plus uniformément possible). L'objectif de bien répartir les variables d'entrée est d'explorer la totalité de l'espace des variables avec un nombre d'échantillons raisonnable. Donc, il faut obtenir des plans efficaces avec un faible nombre d'expériences.

Généralement, les échantillons aléatoires sont générés par un code pseudo aléatoire. L'inconvénient de ces échantillons distribués au hasard est qu'ils ne peuvent pas toujours donner un bon plan pour créer un modèle de substitution précis [Santner *et al.*, 2003] [Forrester *et al.*, 2008]. De ce fait, différents plans d'expériences ont été récemment proposés dans la littérature [Sack *et al.*, 1989] [Simpson *et al.*, 2001] [Karel *et al.*, 2011].

Dans ce qui suit, nous présentons deux méthodes de plans d'expériences qui sont les techniques les plus utilisées : le plan factoriel [Cavazzuti, 2012], le plan hypercube latin [McKay *et al.*, 1979].

1.4.2.1. Les plans d'expériences classiques

Les plans d'expériences classiques [Cavazzuti, 2012] ont été initialement proposés pour remplacer les plans aléatoires. Les limites de variation de chaque variable doivent être définies avant la génération du plan d'expériences. Le plan le plus connu est le plan factoriel complet à deux niveaux par variable.

Le plan factoriel complet est généralement utilisé pour des problèmes qui ont un faible nombre de variables. Néanmoins, quand nous augmentons le nombre de variables de conception, il devient impossible d'utiliser des plans complets [Cavazzuti, 2012].

1.4.2.2. Les plans hypercubes latins

Le plan hypercube latin (LHS) est la méthode la plus populaire des méthodes de remplissage de l'espace, surtout pour la création des modèles de substitution [McKay *et al.*, 1979] [Hoare *et al.*, 2008]. Ce type de plan a été introduit pour faire un choix sélectif des échantillons. Pour la création de ce plan d'expériences, l'espace de conception est divisé en petits carrés de même taille, un carré contient un seul point qui est placé aléatoirement.

Le nombre de points échantillonnés est un critère important pour construire un modèle de substitution précis. En général, plus le nombre de points échantillonnés augmente, plus on a d'informations. C'est l'une des caractéristiques pour améliorer la précision d'un modèle de substitution, mais elle est fréquemment inexécutable à cause du temps de calcul. D'autre part, pour les fonctions d'ordre réduit, l'augmentation du nombre de points échantillonnés contribue faiblement à l'amélioration de la précision du modèle de substitution.

1.4.3. Les principales techniques de métamodélisation

Une fois que la base de données initiale est générée, il reste à choisir la technique du modèle de substitution. Il existe une variété de modèles de substitution dans la littérature. Trois principales techniques sont les plus utilisées : le modèle polynomial [Forrester *et al.*, 2008], les Fonctions à Bases Radiales (RBF) [Simpson *et al.*, 2001] et le modèle de *krigeage* [Sack *et al.*, 1989].

1.4.3.1. Le modèle polynomial

Le modèle polynomial [Forrester *et al.*, 2008] est souvent utilisé pour la construction du modèle de substitution. Il est fréquemment utilisé, il est simple et peut être rapidement construit. Ce modèle est obtenu à l'aide des méthodes de régression linéaire et non linéaire. Un modèle polynomial d'ordre deux peut être exprimé sous la forme suivante :

$$y = \beta_0 + \sum_i \beta_i X_i + \sum_{i < j} \beta_{ij} X_i X_j + \sum_i \beta_{ii} X_i^2 \quad (1.14)$$

Ce type de modèle permet l'estimation d'une surface de réponse pour étudier les effets linéaires, les effets quadratiques et les effets d'interaction :

- y représente la fonction de réponse,
- β_0 est la constante polynomiale qui exprime l'effet moyen général,
- β_i , β_{ii} et β_{ij} sont les coefficients des effets linéaires, quadratiques et interaction respectivement,
- X_i et X_j représentent les variables codées indépendantes.

Ce modèle est l'une des techniques les plus utilisées dans la littérature [Gu, 2001] [Forrester *et al.*, 2008] [Raymond *et al.*, 2016], on trouve trois cas du modèle polynomial : le polynôme quadratique, le polynôme linéaire et le polynôme cubique. Le polynôme quadratique est le plus populaire puisqu'il permet d'approximer une réponse non-linéaire.

1.4.3.2. Les fonctions à bases radiales

La technique des fonctions à bases radiales '*Radial Basis Functions*' (RBF) est une technique d'interpolation en grandes dimensions de données dispersées. Le RBF utilise une approche d'interpolation classique basée sur la distance entre les points supports [Simpson *et al.*, 2001]. Cette approximation s'écrit sous la forme [Carr *et al.*, 1997] :

$$f(x) = \sum_{i=1}^m \omega_i \varphi(\|X - X_i\|) \quad (1.15)$$

où ω_i sont les fonctions poids, φ est une fonction radiale et $\| \cdot \|$ représente la norme euclidienne. Les fonctions radiales les plus utilisées sont illustrées dans le tableau suivant [Carr *et al.*, 1997]:

Tableau 1.1 : les fonctions radiales les plus utilisées avec $r = \|X - X_i\|$

Fonction	$\varphi(r)$
Gaussienne	e^{-r^2}
Multiquadrique	$\sqrt{1+r^2}$
Thin plate spline	$R^2 \ln(r)$
Wendland	$(1-r)^3(3r+1)$

1.4.3.3. La technique de krigeage

La technique de *krigeage* ‘*Kriging*’ a été initialement proposée par Danie Gerhardus Krige en géostatistique [Sack *et al.*, 1989]. Elle a été étendue sur plusieurs domaines tels que l'électromagnétisme, les sciences de l'environnement, la métrologie et le domaine de la synthèse des circuits analogiques [Jourdan, 2005]. Dans cette approche, la réponse d'un simulateur déterministe est sous la forme suivante [Sack *et al.*, 1989] :

$$Y(x) = \mu(x) + Z(x) \quad (1.16)$$

avec $x \in \mathbb{R}^n$, $\mu(x)$ est la structure déterministe pour l'espérance (pour déterminer le type de *krigeage*). En outre, $Z(x)$ est une fonction aléatoire stationnaire, d'espérance nulle, qui permet à ce modèle d'interpoler tous les points supports.

La technique de *krigeage* est un modèle stochastique avec une covariance exprimée sous la formule suivante [Sack *et al.*, 1989] :

$$\text{Cov}[Z(x(i)), Z(x(j))] = S^2 \cdot R[x(i), x(j)] \quad (1.17)$$

où R est la matrice de corrélation et S^2 la variance, $R(x(i), x(j))$ est la fonction de corrélation entre deux points supports $x(i)$ et $x(j)$.

La prédiction \hat{y} de Y est estimée par la méthode de *krigeage* en minimisant l'erreur quadratique moyenne ‘*mean squared error*’ (MSE).

Le modèle de *krigeage* est un modèle d'interpolation, donc les erreurs des prédictions aux points supports sont nulles. Ce modèle fournit également une estimation de l'erreur quadratique moyenne. Cette estimation peut être utilisée dans un processus d'amélioration du modèle ou directement par un processus d'optimisation.

1.5. Conclusion

Ce chapitre est réparti en trois sections : dans la première section, nous avons présenté un résumé des métaheuristiques classiques (mono-objectif et multiobjectif). Dans la deuxième section, nous avons détaillé les techniques de dimensionnement optimal des circuits analogiques : l'optimisation basée sur les équations et l'optimisation basée sur la simulation. Finalement, dans la section 1.4, nous avons défini les modèles de substitution et les différentes techniques de modélisation.

D'après ce chapitre, nous pouvons conclure que l'optimisation et la simulation des circuits analogiques nécessitent des longues durées dans le cycle de conception et que l'utilisation des modèles de substitution pourra nous aider à réduire le temps de ce cycle de conception.

Dans le chapitre suivant, nous proposerons l'adaptation et l'application d'un algorithme d'optimisation basé sur l'utilisation du modèle de substitution nommé '*Efficient Global Optimization*' (EGO) pour le dimensionnement/optimisation des circuits analogiques.

Chapitre 2. La technique d'optimisation basée sur les modèles de substitution-Efficient Global Optimization (EGO)

2.1. Introduction

Comme nous l'avons vu dans le chapitre précédent, la phase de conception des circuits analogiques est une tâche compliquée mais, les méthodes classiques de dimensionnement s'avèrent insuffisantes.

Dans ce travail, nous nous intéressons à l'adaptation et au développement de l'algorithme « *Efficient Global Optimization* » (EGO) pour le dimensionnement/optimisation des circuits analogiques. Cette nouvelle approche est récemment proposée dans la littérature, vu qu'elle combine la modélisation et l'optimisation à la fois et elle permet aussi la génération de modèles précis et rapides.

Dans ce chapitre, nous présentons le principe de cette approche d'optimisation, puis nous allons l'appliquer sur des fonctions mathématiques dans le but de la valider. Ainsi, nous allons comparer cette méthode avec deux métaheuristiques (PSO et GA) basées sur la technique de *krigeage*. Cette comparaison est réalisée et analysée en utilisant la métrique « *Wilcoxon Signed Rank Test* ». Ensuite, nous nous proposons d'appliquer l'algorithme EGO aux problèmes de dimensionnement d'un circuit analogique. Nous considérerons l'amplificateur opérationnel à transconductance (OTA). Cet amplificateur est aussi testé avec deux algorithmes (PSO et BSA) qui sont basés sur la simulation. Des comparaisons et des évaluations seront présentées dans ce chapitre.

2.2. L'algorithme EGO

L'algorithme EGO « *Efficient Global Optimization* » est un algorithme d'optimisation assisté par un modèle de substitution, initialement proposé par Donald R. Jones en 1998 [Jones *et al.*, 1998]. Cette approche a pour but de réduire le nombre d'appels à la fonction coûteuse. Elle consiste à utiliser l'erreur d'estimation fournie par le modèle de substitution afin d'enrichir de manière séquentielle le plan d'expériences avec de nouveaux points. La procédure de l'algorithme EGO consiste à parcourir l'espace de recherche et tenter d'améliorer l'objectif à

chaque itération à travers un critère d'enrichissement appelé critère d'amélioration espérée « *Expected Improvement* » (EI) [Jones *et al.*, 1998]. L'algorithme EGO est basé sur l'utilisation de la technique de *krigeage*. Cet algorithme permet d'ajouter les points d'amélioration à ce modèle pendant le processus d'optimisation à travers le critère (EI). Afin d'améliorer la qualité de l'optimum, le processus de cet algorithme maximise le critère (EI). L'étape d'optimisation est réalisée en appliquant l'algorithme Evolution Différentielle (DE) [Mallipeddi *et al.*, 2015].

L'algorithme EGO est basé sur l'enrichissement séquentiel d'un modèle de *krigeage*. Il permet d'ajouter des points supplémentaires à ce modèle à chaque itération à travers le critère EI [Zhan *et al.*, 2017].

Le pseudo-code de l'algorithme EGO est présenté dans l'algorithme 2.1.

Algorithme 2.1 : Pseudo-code de l'EGO

Créer un plan d'expérience initial : $X = [x_1 ; \dots ; x_n]$

Évaluer la fonction en X et $Y = f(X)$.

Calculer f_{\min} le minimum des résultats d'évaluations effectuées

Tant que le critère d'arrêt n'est pas satisfait **Faire**

Construire un modèle de *krigeage* (Calculer la prédiction)

$X_{n+1} \leftarrow$ maximiser EI ($\max EI(x)$) et ajouter x_{n+1} à X .

Évaluer $Y_{n+1} \leftarrow f(x_{n+1})$ et ajouter y_{n+1} à Y .

$Y_{\min} \leftarrow \min(Y)$ (calculer le minimum y_{\min})

$x_{\min} \leftarrow x \in X: y(x) = y_{\min}$

Ré-estimer les paramètres et mettre à jour le modèle de *krigeage*

Fin Tant que

La figure 2.1 illustre le fonctionnement de l'algorithme EGO. La première étape consiste à la construction du modèle de *krigeage* à partir d'un plan d'expériences initial. La deuxième étape a pour but de maximiser le critère d'amélioration espérée (EI), puis, ce critère permet d'ajouter un nouveau point aux données initiales. La dernière étape est la reconstruction du modèle avec le point supplémentaire et le retour à la deuxième étape. Le processus sera réitéré jusqu'à satisfaction d'un critère d'arrêt.

2.2.1. Le critère d'amélioration espérée

L'algorithme EGO enrichit le plan d'expérience à chaque itération avec un nouveau point d'amélioration. Ce point est choisi en fonction du critère d'amélioration espérée (*Expected Improvement* : EI) qui a été proposé par Schonlau [Schonlau, 1997].

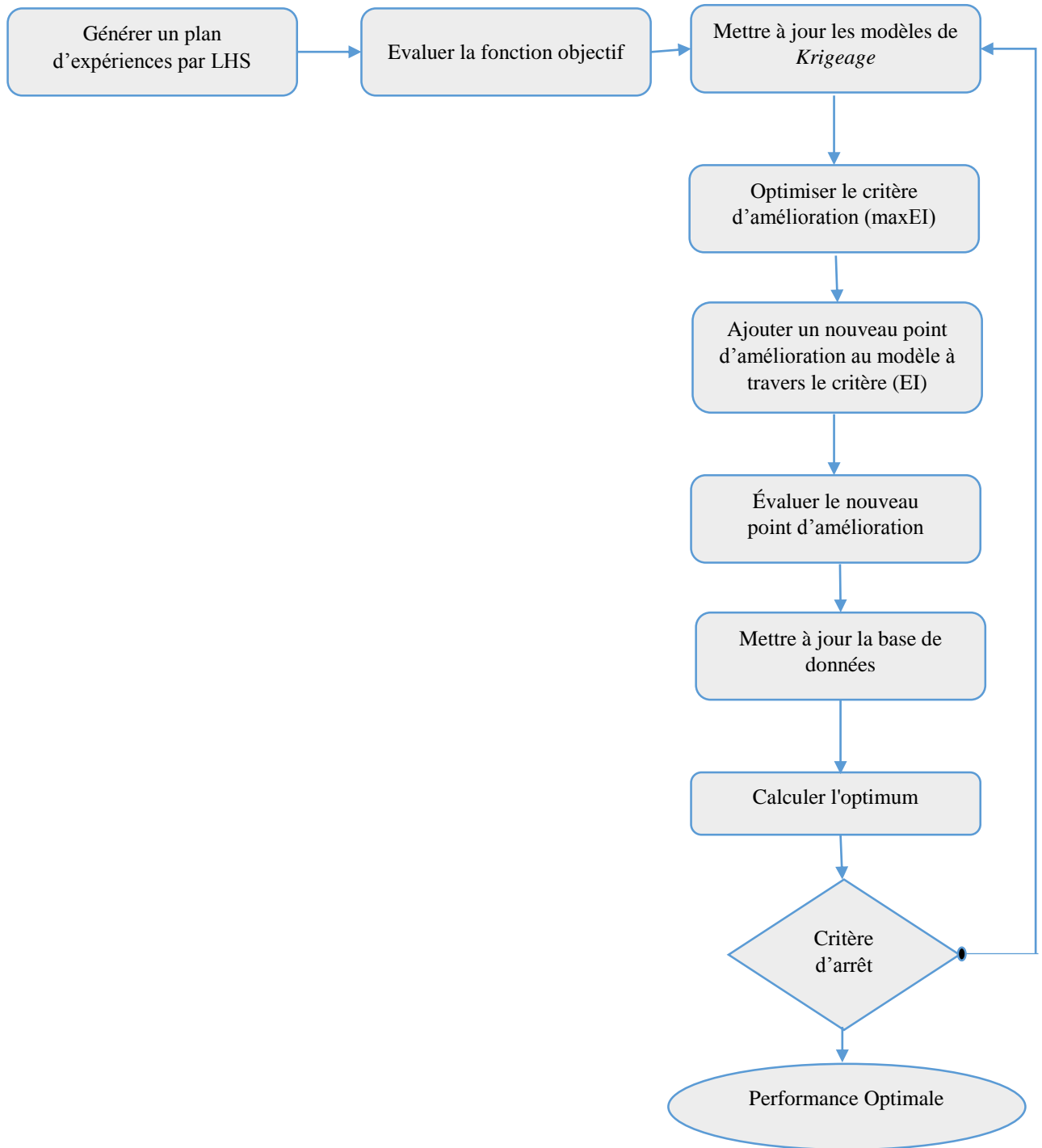


Figure 2.1 : Organigramme de fonctionnement de l'algorithme EGO

Un arbitrage entre exploration du domaine et exploitation des zones connues intéressantes, se fait à travers ce critère.

En utilisant le processus gaussien défini par l'équation de *krigeage* (1.16) présentée dans le chapitre précédent, l'amélioration d'un point x peut être exprimée comme suit :

$$I(x) = \max(y_{\min} - Y(x), 0) \quad (2.1)$$

où y_{\min} est la valeur minimale connue de Y , $s^2(x)$ est la variance et la prédiction $\hat{y}(x)$ de Y est estimée par la technique de *krigeage* en minimisant l'erreur quadratique moyenne (MSE).

Par conséquent, le critère d'amélioration espérée est défini par :

$$EI(x) = (y_{\min} - \hat{y}(x)) \cdot \Phi\left(\frac{y_{\min} - \hat{y}(x)}{s(x)}\right) + s(x) \cdot \phi\left(\frac{y_{\min} - \hat{y}(x)}{s(x)}\right) \quad (2.2)$$

où ϕ et Φ sont respectivement la densité et la fonction de répartition de la loi normale centrée réduite et $s(x)$ est la racine carrée de la prédiction de *krigeage*.

Le premier terme de l'équation (2.2) est grand lorsque la prédiction $\hat{y}(x)$ est plus petite que y_{\min} , ce qui entraîne l'exploitation de la recherche locale (elle sera proche du meilleur point). Par conséquent, le second terme de la fonction EI est petit, lorsque la variance $s(x)$ est grande, ce qui conduit à l'exploration globale. Afin d'optimiser ce critère, l'algorithme métaheuristique "Evolution Différentielle" (DE) est utilisé.

Dans la section suivante, nous appliquons l'algorithme EGO à des fonctions tests et nous le comparons avec d'autres métaheuristicques basées sur la technique de *krigeage*.

2.3. Validation et comparaison avec des fonctions mathématiques

Afin d'évaluer l'algorithme EGO, vingt fonctions tests ont été utilisées, notées de F1 à F20, à différentes dimensions. Ces fonctions mathématiques sont présentées en annexe. Nous testons cet algorithme tout en calculant l'erreur relative de chaque fonction.

L'algorithme EGO est exécuté avec 100 itérations d'évaluation et un plan d'expériences initial de taille 10d (avec d la dimension du problème) et il a été généré en utilisant la technique du Latin Hypercube (LHS).

Nous avons choisi pour le modèle *krigeage* les paramètres suivants :

- ✓ Type de *krigeage* = ordinaire
- ✓ Modèle de régression : polynôme d'ordre 0
- ✓ Modèle de corrélation : gaussienne

L'algorithme DE est exécuté avec les paramètres suivants :

- ✓ Taille de la population : N =50
- ✓ Nombre d'itérations : 100
- ✓ Facteur de mutation (*differential weight*) : F=0.8
- ✓ Facteur de croisement (probabilité de croisement) : CR =0,8

✓ Stratégie de mutation : Rand/1

Le tableau 2.1 présente les résultats obtenus de 20 fonctions de test avec l'algorithme EGO.

D'après le tableau, nous constatons que les erreurs relatives aux tests sont faibles pour 18 des 20 fonctions (l'erreur relative tend vers 0) donc on peut conclure que les résultats de l'optimisation avec l'algorithme EGO sont relativement proches des résultats théoriques.

Dans ce qui suit, nous comparons l'algorithme EGO avec deux autres algorithmes d'optimisation basés sur la technique de *krigeage* afin de montrer l'efficacité et la précision de notre approche. Cette comparaison se base sur l'utilisation de la méthode statistique « *Wilcoxon Signed Rank Test* » [Civicioglu, 2013].

2.3.1. La métrique "Wilcoxon Signed Rank Test"

Wilcoxon Signed Rank Test est un test non paramétrique destiné à évaluer la différence entre deux traitements ou entre des conditions dans lesquelles les échantillons sont corrélés.

Nous pouvons présenter le principe des tests non-paramétriques pour la comparaison des deux échantillons indépendants comme suit :

Soit deux ensembles d'observation x_1, \dots, x_n et y_1, \dots, y_n de n observations indépendantes d_1, \dots, d_n tel que $d_i = x_i - y_i$.

A partir d'une distribution inconnue, nous testons le 'Null Hypothesis H_0 ': la distribution inconnue est symétrique autour de 0.

Soit H_a alternative hypothèses 2 possibilités :

- H_a : La distribution inconnue donne principalement des observations positives,
- H_a : La distribution inconnue donne principalement des observations négatives.

A chaque observation d_i est donné un rang qui est l'un des nombres 1, 2, n.

A l'observation de la plus petite valeur numérique est affecté le Rang 1, à l'observation de la deuxième plus petite valeur numérique est attribué le Rang 2, etc.

On doit mentionner que l'observation de valeur zéro n'aura pas de rang.

T_+ est égale la somme des Rangs positifs ($T_+ = \sum$ rangs correspondant à d_i positif)

T_- est égale la somme des Rangs négatifs $T_- = \sum$ rangs correspondant à d_i négatif

Si H_0 est vrai, alors T_+ et T_- devraient être plus ou moins égales.

Si on teste H_0 vis à vis H_a par rapport à $T = \min\{T_-, T_+\}$:

- H_a : La distribution inconnue donne principalement des observations positives.

H_0 est rejeté si T_- est négligeable ($T_- < n * \alpha$) ; α est le niveau de signification avec

$\alpha = 0,001$ si $0 < n \leq 10$

$\alpha = 0,01$ si $11 \leq n \leq 60$

$\alpha = 0,05$ si $n \geq 61$

Tableau 2.1 : Validation de l'algorithme par des fonctions tests

	Résultat théorique (F(x))	Résultats de l'optimisation de l'EGO (F'(x))	Valeurs des variables	Erreur Relative (%)
F1	-1.0316	-1.0316	X ₁ =-0.0927 X ₂ = 0.7116	0
F2	3	3	X ₁ =-0.0002 X ₂ = -1.0008	0
F3	-0.3979	-0.3979	X ₁ =3.1401 X ₂ = 2.2783	0
F4	-3.86278	-3.8627	X ₁ =0.1114 X ₂ = 0.5552 x ₃ =0.8530	0
F5	-1	-1	X ₁ =-6.1745 X ₂ = 11.6215	0
F6	-1.8013	-1.8409	X ₁ =2.0694 X ₂ = 1.5708	2.2
F7	0	0	X ₁ =3.0086 X ₂ = 0.5020	0
F8	0	4.0452e-05	X ₁ =0.9944 X ₂ = 0.9892	0,004
F9	-1.9133	-1.9132	X ₁ =-0.5478 X ₂ = -1.5466	0.005
F10	-959.6407	-959.6375	X ₁ =-512.0 X ₂ = 404.18	0.00033
F11	-19.2085	-19.1999	X ₁ =8.0842 X ₂ = 9.6691	0.0447
F12	0	0	X ₁ =0.0166 X ₂ = 0.0050	0
F13	-6.0207	-6.0207	X ₁ =0.7572	0
F14	0	0	X ₁ =-0.0007 X ₂ = 0.0038	0
F15	-2.0626	-2.0625	X ₁ =1.3833 X ₂ = 1.3658	0
F16	-1	-0.9361	X ₁ =0.0679 X ₂ = 0.5176	6.3
F17	0	0	X ₁ =0.9955 X ₂ = 0.8899	0
F18	0	0	X ₁ =0.9976 X ₂ = 2.9936	0
F19	0	0	X ₁ =-0.0098 X ₂ = 0.0055	0
F20	0	0	X ₁ =-4.8499 X ₂ = 8.5890	0

- H_a : La distribution inconnue donne principalement des observations négatives.

H_0 est rejeté si T_+ est négligeable ($T_+ < n * \alpha$).

Si on ne teste pas H_0 contre l'une de ces deux hypothèses, donc on la rejette si minimum $T := \min \{T_+, T_-\}$ est négligeable ($T < n * (\alpha/2)$)

Notre interprétation sera basée sur la distribution normale standard « *p-value* » (p-value ne dépasse pas 1). P-value n'est pas la probabilité que H_0 est vrai mais c'est plutôt la quantité de preuves contre H_0 . En effet, si p-value est plus petit que α , alors on rejette H_0 .

Nous avons utilisé le logiciel SPSS pour effectuer les calculs de « *Wilcoxon Signed Rank test* ».

'*Statistical Package for the Social Sciences*' (SPSS) : est un logiciel d'analyse prédictive SPSS qui permet de prévoir avec fiabilité les évènements à venir et donc de prendre des décisions plus intelligentes, de résoudre les problèmes et d'améliorer les résultats [Nie *et al.*, 1975].

2.3.2. Résultats

Dans cette section, nous présentons les avantages de l'algorithme EGO par rapport aux algorithmes basés sur les modèles de substitution.

Nous effectuons les mêmes tests que l'algorithme EGO pour les deux algorithmes PSO et GA basés sur la technique de *krigeage* sur vingt fonctions de tests et avec les mêmes paramètres, à l'exception des valeurs pour :

- Les paramètres de l'algorithme PSO :
 - ✓ Coefficient d'accélération $c1=1.6$
 - ✓ Coefficient d'accélération $c2=1.6$
 - ✓ Coefficient de constriction $C=0.9$
 - ✓ $w = (\text{nombre de générations} - \text{nombre d'itérations}) / \text{nombre de générations}$.
- Les paramètres de l'algorithme GA :
 - ✓ Taux de mutation=0.1
 - ✓ Taux de croisement=0.5

Nous comparons ces approches en utilisant la métrique « *Wilcoxon Signed Rank Test* ». Les trois approches sont exécutées 100 fois pour les 20 fonctions de tests tout en calculant l'erreur relative correspondante. Ensuite, nous avons utilisé ces résultats comme des vecteurs de taille 100 pour réaliser le *Wilcoxon Test*, en conséquence notre α sera égal à 0.05.

Le tableau 2.2 présente les résultats des comparaisons par paires des trois algorithmes (PSO-*krigeage* vs EGO et GA-*krigeage* vs. EGO) en utilisant la métrique *Wilcoxon Signed Rank Test*.

Tableau 2.2 : Résultats de la comparaison pour les 20 fonctions de tests en utilisant Wilcoxon Rank Test

	PSO-krigeage / EGO				GA-krigeage / EGO			
	<i>p-value</i>	<i>T+</i>	<i>T-</i>	<i>winner</i>	<i>p-value</i>	<i>T+</i>	<i>T-</i>	<i>winner</i>
<i>F1</i>	0	0	4005.0	+	0	0	4656.0	+
<i>F2</i>	0	0	5050.0	+	0	0	5050.0	+
<i>F3</i>	0	134.0	3694.0	+	0	19.0	5031.0	+
<i>F4</i>	0	0	5050.0	+	0	0	5050.0	+
<i>F21</i>	0	0	5050.0	+	0	0	5050.0	+
<i>F6</i>	0	1062.5	3987.5	+	0	863.0	4187.0	+
<i>F7</i>	0	0	5050.0	+	0	0	5050.0	+
<i>F8</i>	0	30.0	5020.0	+	0	0	5050.0	+
<i>F9</i>	0	990.0	0	-	0	3.0	4462.0	+
<i>F10</i>	0	737.0	4313.0	+	0	939.0	4111.0	+
<i>F11</i>	0	335.0	4715.0	+	0	611.0	4439.0	+
<i>F12</i>	0	80.0	4970.0	+	0	34.5	5015.5	+
<i>F13</i>	1.0	0	0	=	0	0	1431.0	+
<i>F14</i>	0	364.0	4101.0	+	0	94.5	4855.5	+
<i>F15</i>	0	0	5050.0	+	0	20.0	5030.0	+
<i>F16</i>	0	497.0	4553.0	+	0	648.0	4402.0	+
<i>F17</i>	0	0	5050.0	+	0	3.0	5047.0	+
<i>F18</i>	0	3022.0	381.0	-	0	0	5050.0	+
<i>F19</i>	1.0	0	0	=	0	0	4950.0	+
<i>F20</i>	0.6925	2410.0	2640.0	=	0.9671	2513.0	2537.0	=
+/-/=		<i>15/2/2</i>				<i>19/0/1</i>		

Le ‘ = ’ indique les cas où les deux algorithmes sont identiques (et $p\text{-value} > \alpha = 0.05$ donc on ne rejette pas l'hypothèse nulle H_0).

Le ‘ + ’ et le ‘ - ’ indiquent des cas dans lesquels H_0 a été rejeté ($p\text{-value} \ll \alpha = 0.05$) et les deux algorithmes ne sont pas identiques. Le signe “+” indique la meilleure performance et le “-” indique la moins bonne performance. La dernière ligne du tableau 2.2 montre la somme totale des trois cas de signification (‘ + ’, ‘ = ’ ou ‘ - ’) pour les deux comparaisons.

D’après ces tableaux, on conclut que pour les deux comparaisons, la valeur de $p\text{-value}$ est supérieure à $\alpha = 0.05$ pour 4 fonctions tests.

La valeur T_+ présentée dans le tableau est égale à la somme des rangs positifs dont les erreurs relatives de l’algorithme EGO sont supérieures aux erreurs relatives de l’algorithme PSO-krigeage et l’algorithme GA-krigeage.

La valeur T_- présentée dans le tableau est égale à la somme des rangs négatifs dont les erreurs relatives de l’algorithme EGO sont inférieures aux erreurs relatives de l’algorithme PSO-krigeage et l’algorithme GA-krigeage.

Nous pouvons conclure que la somme des rangs positifs T_+ est inférieure à la somme des rangs négatifs T_- , par conséquent, les erreurs relatives de l’algorithme EGO sont inférieures aux erreurs relatives des algorithmes PSO-krigeage et GA-krigeage.

Nous constatons que l'erreur relative de l'algorithme EGO est plus petite que celles obtenues via les deux autres algorithmes, donc l'algorithme EGO plus précis.

Dans ce qui suit, nous appliquons cet algorithme au problème de dimensionnement d'un circuit analogique.

2.4. Application et comparaison sur un circuit analogique

2.4.1. Optimisation des performances d'un amplificateur opérationnel à transconductance (OTA)

L'amplificateur opérationnel à transconductance (OTA) est un amplificateur tension-courant dont le courant de sortie est proportionnel à la tension différentielle d'entrée [Hershenson *et al.*, 2001] [H. Daoud *et al.*, 2006].

En 1969, les OTA sont apparus à base de transistor bipolaire [Wheatley *et al.*, 1969]. Après, avec l'évaluation de la technique CMOS, les OTA sont devenus à base de transistor CMOS qui est devenu un élément indispensable dans la conception des circuits analogiques. Les OTA sont présentés dans la littérature sous différentes structures. Le choix dépend des exigences satisfaites lors de la conception [Sanchez-Sinencio, 2002]. Dans ce qui suit, nous considérons le circuit OTA CMOS cascodé replié différentiel de la figure 2.2 puisqu'il montre une meilleure solution avec élimination du bruit en mode commun dans les entrées négatives et positives de l'amplificateur [Bennour *et al.*, 2010].

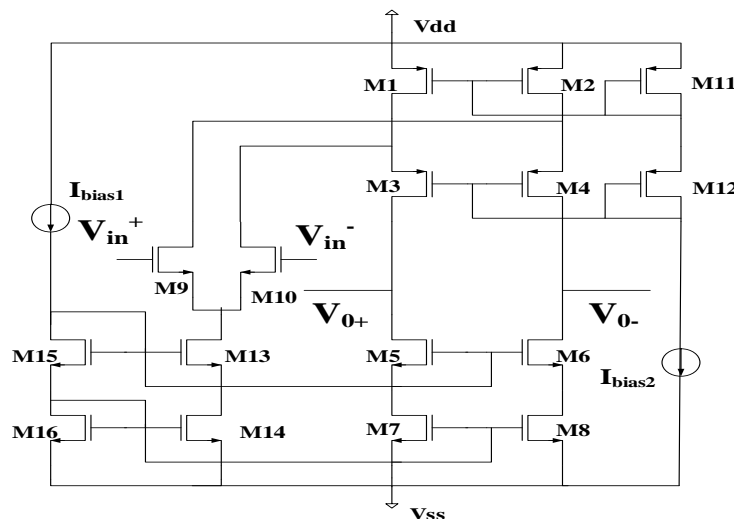


Figure 2.2 : Structure OTA cascodé replié

Le nom « cascodé replié » vient du repliement des transistors cascodés de la paire différentielle de l'amplificateur opérationnel télescopique. Il comporte un étage différentiel,

formé par les transistors (M9-M10) du type NMOS ce qui permet à ce circuit de fournir un gain élevé grâce à la mobilité des transistors NMOS. Le transistor M11 fournit une tension continue de polarisation à la paire des transistors (M1-M2). De même pour le transistor M12 vis-à-vis de la paire des transistors (M7-M8). La source de courant idéale est remplacée par la source de courant cascodé formée par les transistors (M13, M15) et (M14, M16).

Dans ce travail, nous nous intéressons à optimiser trois caractéristiques essentielles de l'OTA qui sont le gain, le taux de rejection du mode commun (CMRR) et le taux de rejection du bruit d'alimentation (PSRR) [Bennour *et al.*, 2010]. L'optimisation du circuit revient à maximiser ces trois fonctions objectifs avec 8 variables. Les variables W et L représentent les largeurs et les longueurs des canaux des transistors CMOS avec un intervalle de variation de L allant de $L_{min} = 0.35 \mu\text{m}$ à $L_{max} = 1.2 \mu\text{m}$ et les W de $[20 \mu\text{m}, 90 \mu\text{m}]$.

Le but de ce test est de comparer notre approche avec deux métaheuristiques (PSO et BSA) basées sur la simulation qui sont déjà présentées dans le premier chapitre. Ces trois algorithmes sont exécutés pour 100 itérations avec une population égale à 100 individus [Drira *et al.*, 2018] [Drira *et al.*, 2019].

2.4.1.1. Maximisation du gain

Nous commençons par maximiser le gain du circuit OTA qui est défini par [Bennour et al, 2010] :

$$A_v = ((V_{o+} - V_{o-}) / (V_{in+} - V_{in-})) \quad (2.3)$$

Nous avons optimisé le gain en tension A_v du circuit OTA considérant 8 variables où les longueurs des transistors sont considérées des variables en plus des largeurs de ceux-ci $W_i \in [20 \mu\text{m}, 90 \mu\text{m}]$ et $L_i \in [0.35 \mu\text{m}, 1.2 \mu\text{m}]$.

Le tableau 2.3 résume les résultats d'optimisation et le temps de calcul du gain de tension.

Tableau 2.3 : Les résultats de simulation du gain pour les trois algorithmes

	Résultats d'optimisation A_v (dB)	Résultats de simulation via H-SPICE A_v (dB)	Erreur relative (%)	Temps d'exécution
<i>EGO</i>	88.971	88.981	<i>0.01</i>	<i>3 min. 10 sec.</i>
<i>PSO-based in-loop</i>	89.677			<i>2 hr. 10 min. 30 sec.</i>
<i>BSA-based in-loop</i>	89.587			<i>2 hr. 0 min. 33 sec.</i>

Le tableau 2.4 présente le dimensionnement optimal du circuit pour tous les résultats trouvés.

Tableau 2.4 : Dimensionnement optimal des transistors du gain

Paramètres	PSO-based in-loop (W/L) (μm)	BSA-based in-loop (W/L) (μm)	EGO (W/L) (μm)
W_1	50.14	90.00	43.15
W_2	90.00	33.74	89.00
W_3	90.00	90.00	77.96
W_4	90.00	20.00	84.65
L_1	1.20	0.84	1.19
L_2	1.02	1.20	0.95
L_3	1.20	1.20	1.19
L_4	1.20	0.35	1.11

La figure 2.3 présente les courbes du gain pour les trois algorithmes.

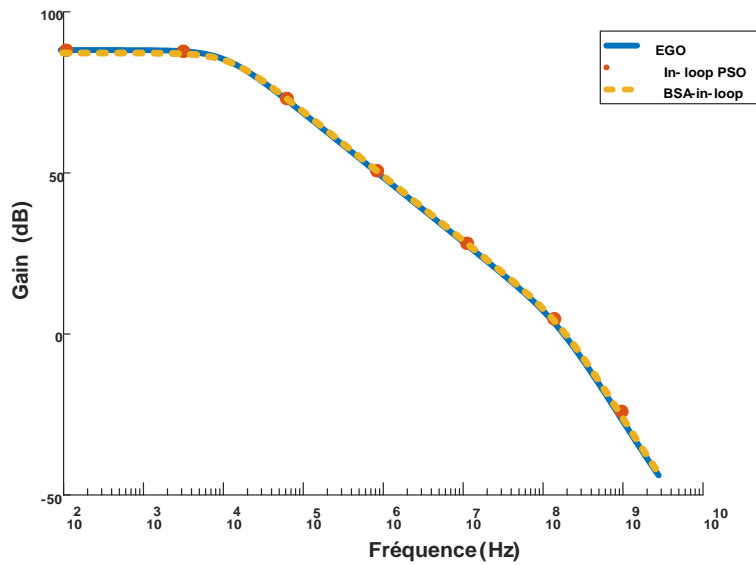


Figure 2.3 : Courbes du gain de l'OTA

Nous remarquons que les résultats de l'algorithme EGO sont proches des résultats obtenus par les algorithmes PSO et BSA qui sont basés sur la simulation. Aussi, nous constatons que notre algorithme converge très rapidement dans un temps d'exécution réduit avec un taux d'erreur n'excédant pas 0.05%.

2.4.1.2. Maximisation du taux de rejection du mode commun

Le CMRR décrit la sensibilité aux changements égaux de tension en deux entrées (positive et négative de l'OTA), et donc la sensibilité de l'amplificateur à rejeter un signal commun. Cette caractéristique est importante dans les applications analogiques où les signaux sont transmis en mode différentiel [Bennour *et al.*, 2010], [Todeschini, 2014] [Dira *et al.*, 2019].

Nous avons maximisé le CMRR du circuit OTA considérant 8 variables de W et L.

Les résultats d'optimisation sont représentés dans le tableau 2.5 suite à l'application des trois algorithmes.

Tableau 2.5 : Les résultats de simulation du CMRR pour les trois algorithmes

	Résultats d'optimisation CMRR (dB)	Résultats de simulation via H-SPICE CMRR (dB)	Erreur relative (%)	Temps d'exécution
<i>EGO</i>	100.157	99.627	0.5	3 min.
<i>PSO-based in-loop</i>	102.190			6 hr. 47 min. 22 sec.
<i>BSA-based in-loop</i>	108.400			6 hr. 37 min. 44 sec.

Le tableau 2.6 montre le dimensionnement optimal du circuit pour tous les résultats du CMRR.

Tableau 2.6 : Dimensionnement optimal des transistors du CMRR.

Paramètres	<i>PSO-based in-loop</i> (W/L) (μm)	<i>BSA-based in-loop</i> (W/L) (μm)	<i>EGO</i> (W/L) (μm)
W_1	20.00	20.00	75.06
W_2	20.00	90.00	20.00
W_3	90.00	42.57	87.72
W_4	90.00	43.74	90.00
L_1	1.20	0.67	1.14
L_2	0.97	0.38	0.94
L_3	0.35	0.42	0.35
L_4	0.35	0.35	0.91

La figure 2.4 représente les courbes du CMRR pour les trois algorithmes.

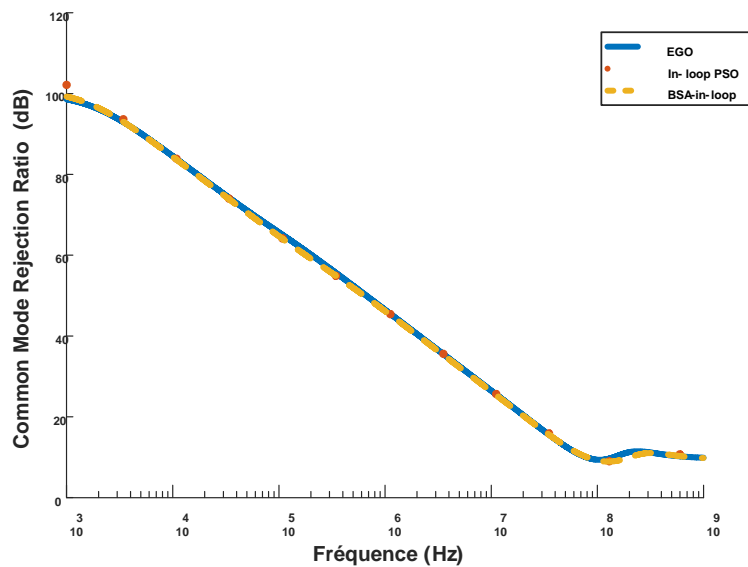


Figure 2.4 : Courbes de CMRR

En observant ces résultats, nous pouvons affirmer que l'algorithme EGO converge très rapidement dans un temps d'exécution réduit avec une erreur relative de l'ordre de 0.5%.

2.4.1.3. Maximisation du taux de rejection du bruit d'alimentation

Le principe de calcul du taux de réjection du bruit d'alimentation repose sur une analyse fréquentielle, suite à l'application de générateurs alternatifs en série avec les sources d'alimentation [Bennour *et al.*, 2010], [Todeschini, 2014].

Le tableau 2.7 présente les résultats d'optimisation et le temps de calcul du PSRR

Tableau 2.7 : Les résultats de simulation du PSRR pour les trois algorithmes

	Résultats d'optimisation PSRR (dB)	Résultats de simulation via H-SPICE PSRR (dB)	Erreur relative (%)	Temps d'exécution
<i>EGO</i>	81.412	80.744	0.8	3 min. 18 sec.
<i>PSO-based in-loop</i>	81.377			7 hr.
<i>BSA-based in-loop</i>	81.239			6 hr. 46 min. 33 sec.

Le tableau 2.8 présente le dimensionnement optimal du circuit pour tous les résultats du PSRR.

Tableau 2.8 : Dimensionnement optimal des transistors du PSRR

Paramètres	<i>PSO-based in-loop</i> (W/L) (μm)	<i>BSA-based in-loop</i> (W/L) (μm)	<i>EGO</i> (W/L) (μm)
W_1	89.68	54.33	60.71
W_2	90.00	20.00	90.00
W_3	90.00	90.00	70.04
W_4	89.99	90.00	90.00
L_1	1.20	1.20	1.07
L_2	0.99	0.35	0.87
L_3	1.20	0.35	1.20
L_4	1.20	1.03	1.20

La figure 2.5 illustre les courbes de PSRR pour les trois algorithmes.

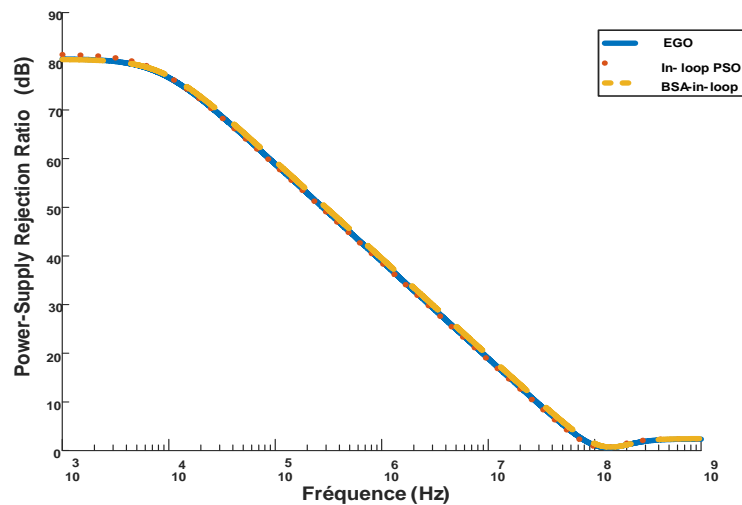


Figure 2.5 : Courbes de PSRR

2.4.2. Récapitulatif et interprétations

Cette étude comparative nous a permis de conclure sur l'efficacité et la rapidité de l'algorithme EGO dans le dimensionnement du circuit analogique OTA cascodé replié. Nous constatons que cette technique nous a donné des résultats proches aux résultats obtenus par les algorithmes PSO et BSA qui sont basés sur la simulation. En plus, elle converge très rapidement dans un temps d'exécution réduit avec une petite plage d'erreurs relatives pour les différentes performances.

2.5. Conclusion

Dans ce chapitre, nous avons introduit l'algorithme EGO. C'est un algorithme d'optimisation assisté par un modèle de substitution. La possibilité d'utiliser un modèle de substitution à la place du modèle réel peut avoir l'avantage de diminuer le nombre d'évaluations coûteuses de la fonction objectif. Le but principal de cette méthode est de rajouter des points au plan d'expériences au niveau des zones optimales du problème, afin que ses solutions soient connues avec la plus grande précision possible. Dans ce chapitre, nous avons testé et validé cet algorithme avec 20 fonctions de test et nous avons effectué une comparaison avec des algorithmes basés sur un modèle de substitution dans lesquels l'efficacité et la robustesse de l'algorithme EGO sont montrées. Puis, nous avons appliqué cet algorithme aux problèmes de dimensionnement d'un circuit analogique, à savoir, l'amplificateur opérationnel à transconductance CMOS. Ce circuit a été aussi testé avec des algorithmes métaheuristiques qui sont basés sur la simulation. Nous avons pu remarquer que l'algorithme EGO converge très rapidement dans un temps d'exécution réduit avec un taux d'erreur négligeable. Les résultats de ce travail ont été publiés dans une communication de conférence [Drira et al. 2018] et un article de revue [Drira et al. 2019].

Cependant, cela nous a alors amenés à tester cette méthode avec d'autres circuits analogiques, et à améliorer ce fonctionnement afin de l'appliquer à des problèmes multiobjectif. Nous présentons ces évaluations dans le chapitre qui suit.

Chapitre 3 : Amélioration de l'algorithme EGO :

rapidité et précision

3.1. Introduction

Dans le chapitre précédent, nous avons introduit l'algorithme EGO. En particulier, nous avons comparé sa rapidité et sa précision par rapport aux algorithmes utilisant des métaheuristiques et qui sont basés sur la simulation pour la résolution des problèmes de dimensionnement des circuits analogiques. Les résultats obtenus sont très encourageants. C'est ainsi qu'une analyse des performances de cet algorithme fut nécessaire, qui sera le but de ce chapitre.

Ainsi, dans ce chapitre, nous présentons, nous analysons et nous proposons une amélioration des performances de cet algorithme. Ce chapitre est composé de trois sections. Dans la première section, nous comparons la précision de l'algorithme EGO par rapport aux autres algorithmes via des métaheuristiques et basés sur la technique de métamodélisation. Cette comparaison est réalisée pour deux circuits analogiques en utilisant la métrique '*Wilcoxon Signed Rank Test*'.

Dans la deuxième section, nous effectuons une étude et une adaptation du critère « *Pseudo expected improvement* » (PEI) dans l'algorithme EGO pour la conception des circuits analogiques. Ce nouveau critère est proposé pour développer un algorithme EGO parallèle. À chaque itération, ce dernier permet de produire plusieurs points pour le calcul parallèle. Cette adaptation apporte une meilleure efficacité et une meilleure rapidité à l'algorithme EGO.

Dans la dernière section, nous proposons l'adaptation de cette approche pour la résolution des problèmes multiobjectif. En effet, une nouvelle approche basée sur le critère « *Expected Improvement Matrix* » (EIM) a été développée. C'était dans le but de réduire le temps de calcul des critères (EI) lors de la manipulation de problèmes multiobjectif.

3.2. Influence de la taille de la base de données

L'EGO est une approche d'optimisation basée sur l'utilisation d'un modèle de substitution. Il permet de réduire le nombre d'appels coûteux de la fonction de performance en estimant les zones intéressantes de l'espace à partir du modèle de *krigeage* [Zhan *et al.*, 2017]. Comme déjà présenté dans le chapitre précédent, nous avons effectué une comparaison entre notre approche et des algorithmes métaheuristiques basés sur le modèle de substitution *krigeage* sur des

fonctions tests. Nous avons remarqué que l'algorithme EGO nous a permis d'obtenir des résultats précis par rapport aux autres algorithmes.

L'algorithme EGO ajoute un point d'échantillonnage à chaque itération au plan d'expériences initial tout en mettant à jour le modèle de *krigeage*. Alors, nous minimisons la base de données initiale du modèle et nous comparons cette approche avec deux métaheuristiques basées sur le modèle de substitution *krigeage* sur des circuits analogiques. Cette approche donne une meilleure précision même avec un nombre limité d'échantillons initiaux. Les sous-sections suivantes présentent les résultats de ces travaux.

3.2.1. Application sur des circuits analogiques

Dans ce qui suit, nous testons l'algorithme EGO et deux algorithmes d'optimisation métaheuristiques (PSO et GA) basés sur la technique de *krigeage* sur deux circuits analogiques CMOS, à savoir, le convoyeur de courant et le suiveur de tension.

3.2.1.1. Optimisation des performances du circuit convoyeur de courant

Le convoyeur de courant CC est un circuit actif qui réalise plusieurs fonctions analogiques. Ce circuit peut jouer au même temps le rôle d'un élément de base dans la conception des circuits intégrés puisqu'il simplifie le fonctionnement de plusieurs circuits complexes. Aussi, il aide à la création d'une implémentation simple, avec l'avantage de fonctionner en mode "tension" et en mode "courant". Le circuit de convoyeur de courant de la deuxième génération classe AB (CCII) est présenté dans la figure 3.1 [Sedra *et al.*, 1970] [Sedra *et al.*, 1990].

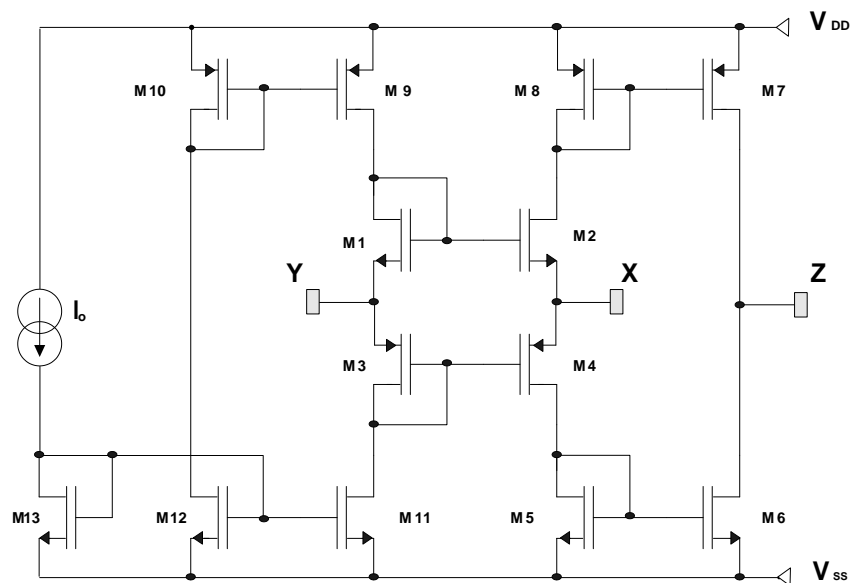


Figure 3.1 : Schéma d'un CCII+ class AB

Les transistors M1-M4 forment une boucle translinéaire et les transistors M5-M8 forment des miroirs de courant reproduisant au port Z le courant du port X [Aronhime *et al.*, 1978] [Surakamptron *et al.*, 1988].

Les variables du circuit sont les largeurs des transistors NMOS et PMOS 'WN' et 'WP'. Nous considérons que la longueur 'L' des transistors MOS est égale à 0.35µm, VDD=-VSS=2.5V et le courant de polarisation *I*_{bias} est égale à 100µA. La technologie utilisée est AMS CMOS 0.35µm.

Dans cette partie, les principales performances à optimiser du circuit convoyeur de courant (CCII) [Sedra *et al.*, 1970] sont les suivantes : premièrement minimiser la résistance parasite au port X (R_x) et deuxièmement maximiser la fréquence de coupure en courant (F_{ci}). Nous appliquons les trois algorithmes d'optimisation sur le circuit CCII. Deux bases de données initiales ont été générées. La première base de données contient 5 points d'échantillons et la deuxième est composée de 35 points d'échantillons.

Nous présentons dans les tableaux 3.1 et 3.2 les résultats d'optimisation que nous avons obtenus, suite à 30 itérations des trois algorithmes développés et les résultats de simulation par l'outil HSPICE MT. Ces tableaux montrent les résultats des deux fonctions objectifs du circuit CCII.

Tableau 3.1 : Les résultats de la simulation (R_x) du circuit CCII

	Valeurs des Paramètres (w_1, w_2) (µm)	Résultats optimisation d'algorithmes proposés (R_x) (Ω)	Résultats Simulation (R_x) (Ω) H-SPICE	Erreur Relative (%)
5 points				
EGO	$w_1= 45.38$ $w_2= 79.15$	264.51	264.51	0
GA_Krigeage	$w_1= 45.26$ $w_2= 78.90$	265.72	264.94	0.29
PSO_Krigeage	$w_1=45.35$ $w_2=78.86$	264.84	265.21	0.14
35 points				
EGO	$w_1=46.53$ $w_2=79.16$	262.71	262.71	0
GA_Krigeage	$w_1= 46.65$ $w_2= 79.33$	262.72	262.35	0.141
PSO_Krigeage	$w_1= 45.43$ $w_2= 78.98$	262.71	262.68	0.011

Les figures 3.2 et 3.3 présentent les courbes des deux solutions optimales de la résistance parasite au port X (R_x) et du gain en courant (G_{ci}) avec l'algorithme EGO, respectivement.

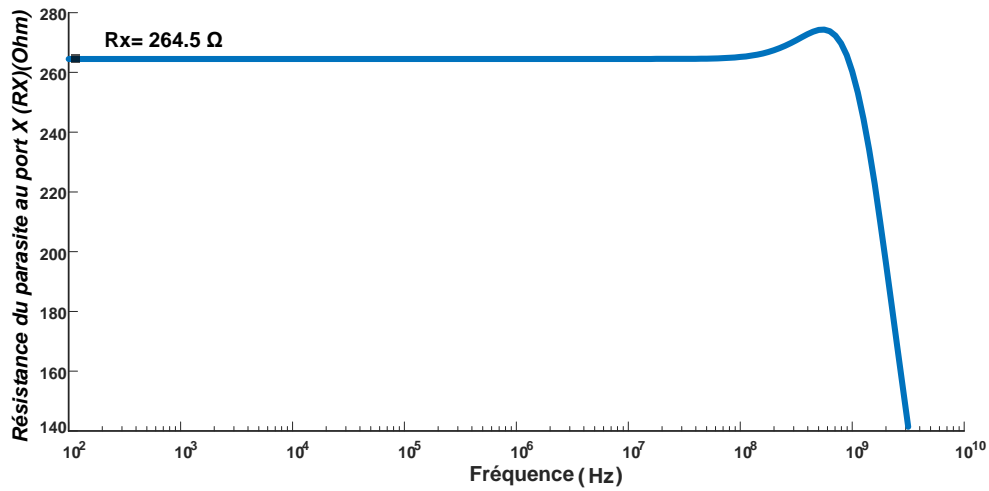


Figure 3.2 : Courbe de la résistance du parasite au port X (R_x) du circuit CCII

Tableau 3.2 : Les résultats de la simulation (F_{ci}) du circuit CCII

	Valeurs des Paramètres (w_1, w_2) (μm)	Résultats optimisation d'algorithmes proposés (F_{ci}) (GHz)	Résultats Simulation (F_{ci}) (GHz) H-SPICE	Erreur Relative (%)
5 points				
<i>EGO</i>	$w_1 = 10.75$ $w_2 = 17.8$	1.0732	1.0732	0
<i>GA_Krigeage</i>	$w_1 = 10.73$ $w_2 = 18.14$	1.0718	1.0690	0.262
<i>PSO_Krigeage</i>	$w_1 = 10.72$ $w_2 = 17.80$	1.0731	1.0736	0.047
35 points				
<i>EGO</i>	$w_1 = 10.75$ $w_2 = 17.80$	1.0732	1.0732	0
<i>GA_Krigeage</i>	$w_1 = 10.61$ $w_2 = 17.77$	1.0731	1.0750	0.177
<i>PSO_Krigeage</i>	$w_1 = 10.75$ $w_2 = 17.79$	1.0731	1.0734	0.028

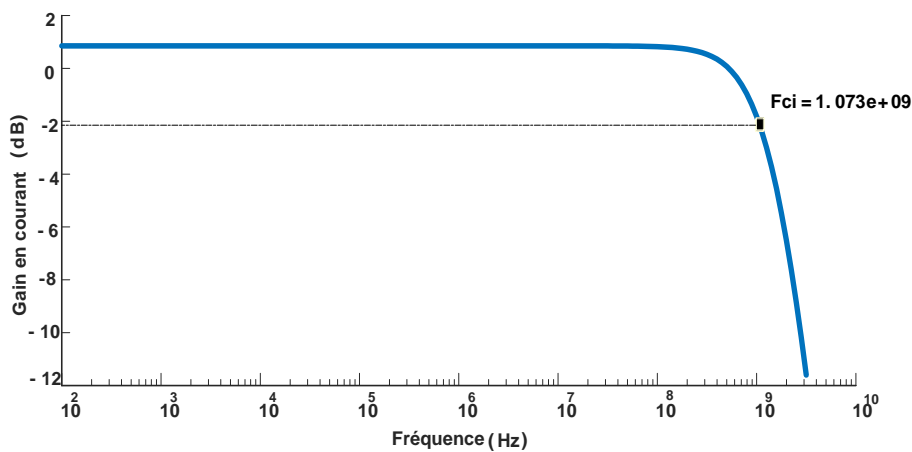


Figure 3.3 : Courbe de gain en courant du circuit CCII

D'après ces tests, les deux algorithmes métaheuristiques donnent une plage d'erreurs entre [0.011 0.29] %. Donc, nous pouvons conclure que l'algorithme EGO permet d'obtenir des résultats plus fiables et une erreur relative nulle pour les deux fonctions objectifs du circuit CCII par rapport aux autres algorithmes, avec une petite base de données initiale et un nombre minimum d'itérations.

3.2.1.2. Optimisation des performances d'un suiveur de tension

Le suiveur de tension est souvent utilisé pour la construction de mémoires tampons pour les circuits logiques. La figure 3.3 présente un circuit d'un suiveur de tension "Voltage Follower" (VF) [Guerra-Gomez *et al.*, 2010] [Tlelo-Cuautle, 2012].

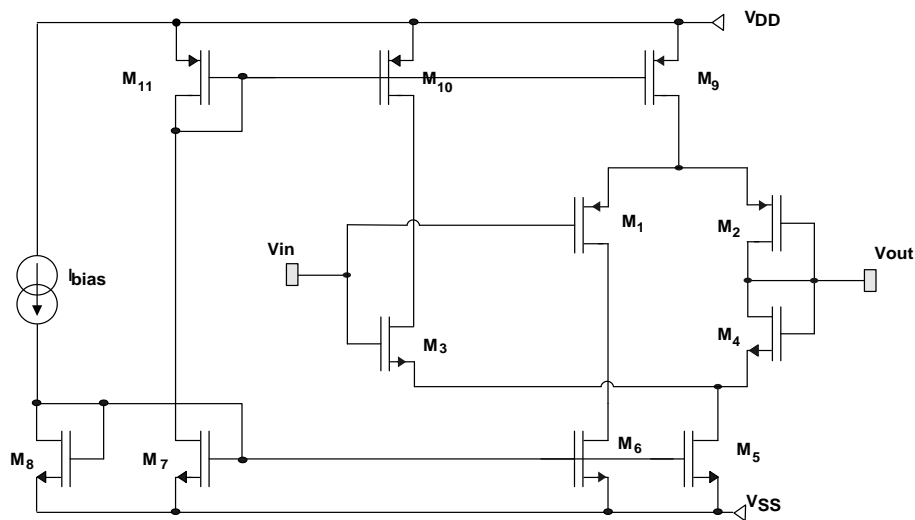


Figure 3.4 : Schéma du circuit d'un suiveur de tension

Les variables du circuit sont présentées dans le tableau 3.3. Nous considérons que la longueur du canal des transistors MOS 'L' est égale à $0.35\mu\text{m}$, $V_{DD} = -V_{SS} = 1.5\text{V}$ et $I_{bias} = 50\mu\text{A}$. La charge capacitive utilisée pour chaque simulation est égale à 1pF . La technologie utilisée est CMOS $0.35\mu\text{m}$.

Tableau 3.3 : Variables du circuit suiveur de tension

Transistors	Variables
M_9, M_{11}, M_3, M_4	W_1
M_5, M_7, M_8, M_1, M_2	W_2
M_{10}	$0.5 W_1$
M_6	$0.5 W_2$

Les deux fonctions objectifs à optimiser sont la tension d'offset (V_{Offset}) à minimiser et la fréquence de coupure en tension (F_{cv}) à maximiser.

Nous avons appliqué ces algorithmes pour ces deux fonctions objectifs avec deux bases de données initiales. La première base de données contient 5 points d'échantillons et la deuxième est composée de 35 points d'échantillons.

Les tableaux 3.4 et 3.5 montrent les résultats d'optimisation que nous avons obtenus des différents algorithmes pour 30 itérations avec les deux fonctions objectifs du circuit VF.

Tableau 3.4 : Les résultats de la simulation (V_{offset}) du circuit VF

	Valeurs des Paramètres (w_1, w_2) (μm)	Résultats optimisation d'algorithmes proposés (V_{offset}) (V)	Résultats Simulation (V_{offset}) (V) H-SPICE	Erreur Relative (%)
5 points				
<i>EGO</i>	$w_1=15.27$ $w_2= 92.20$	-0.0935	-0.0935	0
<i>GA_Krigeage</i>	$w_1=15.26$ $w_2= 92.22$	-0.0934	-0.0935	0.11
<i>PSO_Krigeage</i>	$w_1=15.26$ $w_2= 89.27$	-0.0935	-0.0936	0.11
35 points				
<i>EGO</i>	$w_1=20.19$ $w_2= 15.95$	-0.0980	-0.0980	0
<i>GA_Krigeage</i>	$w_1=42.19$ $w_2= 16.32$	-0.0962	-0.09624	0.04
<i>PSO_Krigeage</i>	$w_1=65.66$ $w_2= 62.11$	-0.0980	-0.09798	0.01

Les figures 3.5 et 3.6 montrent les courbes des deux solutions optimales de la tension d'offset (V_{offset}) et le gain en tension (G_{cv}) avec l'algorithme EGO.

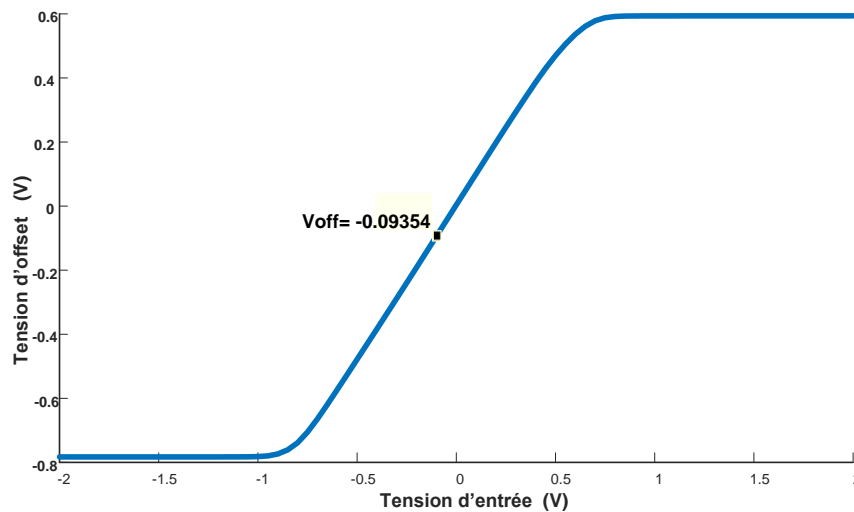
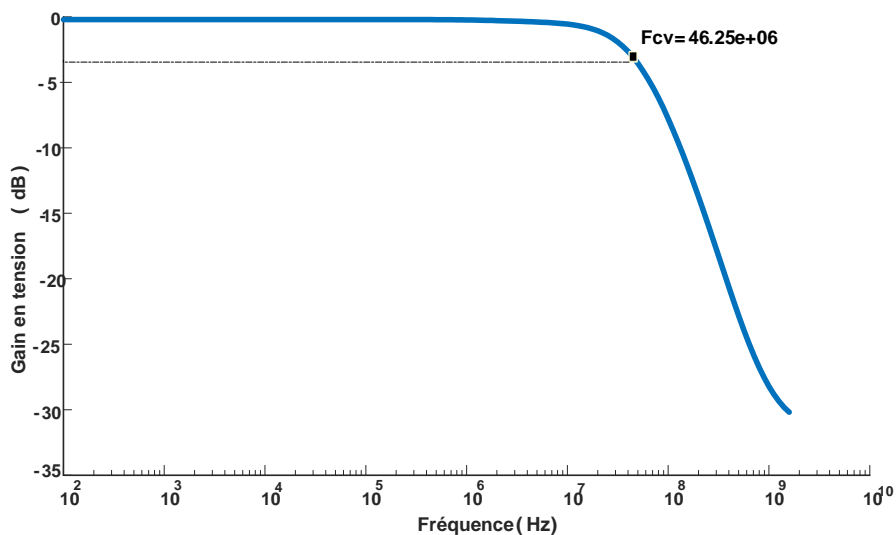


Figure 3.5 : Courbe de l'offset en tension (V_{offset}) du circuit VF

Tableau 3.5 : Les résultats de la simulation (F_{cv}) du circuit VF

	Valeurs des Paramètres (w_1, w_2) (μm)	Résultats optimisation d'algorithmes proposés (F_{cv}) (MHz)	Résultats Simulation (F_{cv}) (MHz) H-SPICE	Erreur Relative (%)
5 points				
<i>EGO</i>	$w_1=53.88$ $w_2=57.19$	46.2480	46.2480	0
<i>GA_Krigeage</i>	$w_1=54.02$ $w_2=57.46$	46.2265	46.2390	0.03
<i>PSO_Krigeage</i>	$w_1=53.86$ $w_2=57.18$	46.2307	46.2570	0.06
35 points				
<i>EGO</i>	$w_1=50.02$ $w_2=83.17$	46.4148	46.4070	0.02
<i>GA_Krigeage</i>	$w_1=57.92$ $w_2=48.20$	46.3919	46.4040	0.03
<i>PSO_Krigeage</i>	$w_1=48.43$ $w_2=47.87$	46.4163	46.4070	0.02

**Figure 3.6 :** Courbe de gain en tension du circuit VF

D'après ces tableaux, nous pouvons conclure que les deux fonctions objectifs aboutissent à des erreurs relatives presque nulles. Néanmoins, les résultats de l'algorithme EGO sont approximativement nuls même avec une petite base de données. Donc, cette approche est plus précise que les deux autres approches basées sur la métamodélisation.

3.2.2. Évaluation et interprétation

Dans ce qui suit, nous présentons une étude comparative entre ces algorithmes en utilisant la métrique statistique « *Wilcoxon Signed Rank Test* » présentée dans le chapitre 2. Nous

appliquons cette comparaison sur les deux circuits analogiques (CCII et VF) traités aux sections 3.2.1.1 et 3.2.1.2.

Au début, nous exécutons 100 fois l'algorithme EGO et les deux métaheuristiques (PSO et GA) basées sur la technique de *krigeage* pour les différentes fonctions objectifs. Ensuite, nous vérifions les résultats trouvés avec un programme en C++ (Visual Studio) qui fait l'appel au simulateur H-SPICE MT.

Dans ce travail, nous nous intéressons à comparer la précision des différents algorithmes. Donc, nous testons ces performances des différents circuits tout en calculant leurs erreurs relatives correspondantes. Ensuite, on va utiliser ces résultats comme des vecteurs de taille 100 pour réaliser *Wilcoxon Test* donc la valeur de α sera égale à 0.05.

Dans ce chapitre, nous comparons les erreurs relatives entre l'EGO et le PSO-*krigeage* ainsi que les erreurs relatives entre l'EGO et le GA-*krigeage* avec différents échantillons initiaux. La première base de données initiale comprend 5 échantillons et la deuxième base de données contient 35 échantillons.

Les tableaux 3.6 et 3.7 montrent les résultats des comparaisons par paires entre les différents algorithmes pour 30 itérations en utilisant la métrique choisie pour les deux circuits et leurs différentes fonctions objectifs.

Tableau 3.6 : Résultats de la comparaison entre les trois algorithmes sur le circuit CCII ($\alpha = 0.05$)

	<i>PSO-krigeage / EGO</i>				<i>GA-krigeage / EGO</i>			
	<i>p-value</i>	<i>T+</i>	<i>T-</i>	<i>winner</i>	<i>p-value</i>	<i>T+</i>	<i>T-</i>	<i>winner</i>
Minimisation de la Résistance (R_x)								
R_x (5 échantillons)	0	0	5050	+	0	0	5050	+
R_x (35 échantillons)	0	0	3403	+	0	0	5050	+
Maximisation de la fréquence de coupure en courant (F_{ci})								
F_{ci} (5 échantillons)	0	0	5050	+	0	0	5050	+
F_{ci} (35 échantillons)	0	0	5050	+	0	0	5050	+
+/-/=	4/0/0				4/0/0			

Le '=' indique les cas où les deux algorithmes sont identiques (et $p\text{-value} > \alpha = 0.05$ donc on ne rejette pas l'hypothèse nulle H_0).

Le '+' et le '-' indiquent des cas dans lesquels H_0 a été rejeté ($p\text{-value} \ll \alpha = 0.05$) et les deux algorithmes ne sont pas identiques. Le signe "+" indique la meilleure performance et le "-"

indique la moins bonne performance. La dernière ligne des tableaux montre la somme totale des trois cas de signification ('+', '=' ou '-') pour les deux comparaisons.

Tableau 3.7 : Résultats de la comparaison entre les trois algorithmes sur le circuit VF ($\alpha = 0.05$)

<i>PSO-krigeage / EGO</i>				<i>GA-krigeage / EGO</i>				
<i>p-value</i>	<i>T+</i>	<i>T-</i>	<i>winner</i>	<i>p-value</i>	<i>T+</i>	<i>T-</i>	<i>winner</i>	
<i>Minimisation de la tension d'offset (V_{Offset})</i>								
V_{offset} (5 échantillons)	0	262	4788	+	0	416	4634	+
V_{offset} (35 échantillons)	0	0	5050	+	0	0	5050	+
<i>Maximisation de la fréquence de coupure en tension (F_{cv})</i>								
F_{cv} (5 échantillons)	0	0	4656	+	0	0	5050	+
F_{cv} (35 échantillons)	0	1885	3165	+	0	2150	2900	+
+/-/=				4/0/0				

D'après ces tableaux, on conclut que la valeur de p-value est égale à 0 pour tous les tests. Cette valeur est inférieure $\alpha = 0.05$. Donc, on rejette l'hypothèse nulle H_0 . Par conséquent, les deux algorithmes ne sont pas identiques.

La valeur T_+ présentée dans les tableaux est égale à la somme des rangs positifs dont les erreurs relatives de l'algorithme EGO sont supérieures aux erreurs relatives de l'algorithme PSO-krigeage et l'algorithme GA-krigeage.

La valeur T_- présentée dans les tableaux est égale à la somme des rangs négatifs dont les erreurs relatives de l'algorithme EGO sont inférieures aux erreurs relatives de l'algorithme PSO-krigeage et l'algorithme GA-krigeage.

Nous pouvons conclure que la somme des rangs positifs T_+ est inférieure à la somme des rangs négatifs T_- , par conséquent, les erreurs relatives de l'algorithme EGO sont inférieures aux erreurs relatives des algorithmes PSO-krigeage et GA-krigeage. L'algorithme EGO est le meilleur par rapport aux autres algorithmes que nous avons testés pour l'optimisation des différentes performances de circuits analogiques.

Dans ce qui suit, nous comparons les trois algorithmes avec un nombre limité de 5 échantillons initiaux pour l'algorithme EGO et 35 échantillons initiaux pour le PSO-krigeage et le GA-krigeage.

Les tableaux 3.8 et 3.9 présentent les résultats que nous avons obtenus relatifs à la comparaison entre les différents algorithmes en utilisant la métrique citée plus haut pour les deux circuits analogiques considérés.

Tableau 3.8 : Résultats de la comparaison entre les trois algorithmes sur le circuit CCII ($\alpha = 0.05$)

	<i>p-value</i>	<i>T+</i>	<i>T-</i>	<i>winner</i>
Minimisation de la Résistance (R_x)				
<i>EGO / PSO-krigeage</i>	0	0	3403	+
<i>EGO / GA-krigeage</i>	0	0	5050	+
Maximisation de la fréquence de coupure en courant (F_{ci})				
<i>EGO / PSO-krigeage</i>	0	0	5050	+
<i>EGO / GA-krigeage</i>	0	0	5050	+
+/-/=		4/0/0		

Tableau 3.9 : Résultats de la comparaison entre les trois algorithmes sur le circuit VF ($\alpha = 0.05$)

	<i>p-value</i>	<i>T+</i>	<i>T-</i>	<i>winner</i>
Minimisation de la tension d'offset (V_{Offset})				
<i>EGO / PSO-krigeage</i>	0	23	5027	+
<i>EGO / GA-krigeage</i>	0	490	4560	+
Maximisation de la fréquence de coupure en tension (F_{cv})				
<i>EGO / PSO-krigeage</i>	0	0	5050	+
<i>EGO / GA-krigeage</i>	0	0	5050	+
+/-/=		4/0/0		

D'après ces tableaux, nous remarquons que la valeur de p-value est égale à 0 pour tous les tests. Cette valeur est inférieure $\alpha = 0.05$. Donc, on rejette l'hypothèse nulle H_0 . Par conséquent, les deux algorithmes ne sont pas identiques.

Comme déjà présenté dans la dernière partie, la valeur T_+ présentée dans les tableaux est égale à la somme des rangs positifs dont les erreurs relatives de l'algorithme EGO sont supérieures aux erreurs relatives de l'algorithme PSO-krigeage et l'algorithme GA-krigeage.

La valeur T_- présentée dans les tableaux est égale à la somme des rangs négatifs dont les erreurs relatives de l'algorithme EGO sont inférieures aux erreurs relatives de l'algorithme PSO-krigeage et l'algorithme GA-krigeage.

Nous avons conclu que la somme des rangs positifs T_+ est inférieure à la somme des rangs négatifs T_- , par conséquent, les erreurs relatives de l'algorithme EGO sont inférieures aux erreurs relatives des algorithmes PSO-krigeage et GA-krigeage. L'algorithme EGO est donc plus précis avec une base de données initiale réduite par rapport à celle utilisée pour les autres algorithmes.

Dans ce qui suit, nous effectuons une présentation par les boxplots pour mettre en relief la robustesse des algorithmes considérés.

Les figures 3.7 et 3.8 présentent les tests de robustesse (100 fois) des erreurs relatives des différents algorithmes des deux circuits.

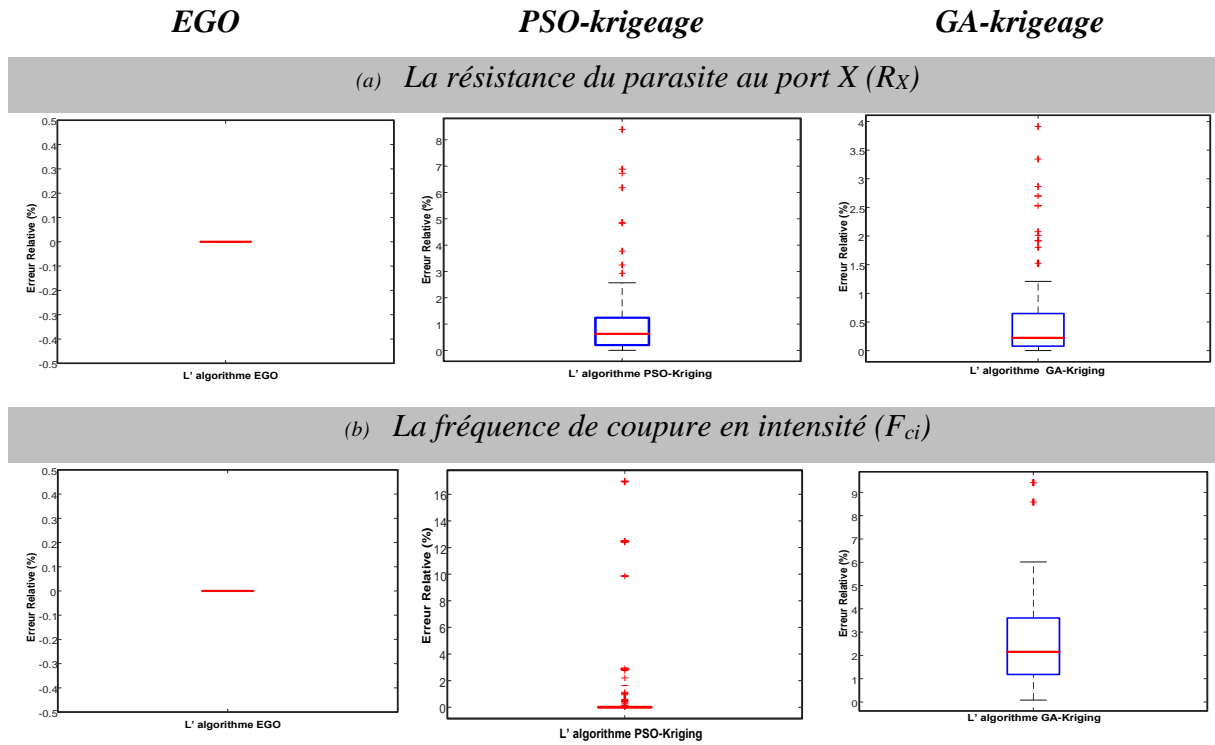


Figure 3.7 : Tests de la robustesse des erreurs relatives pour 100 itérations du circuit CCII

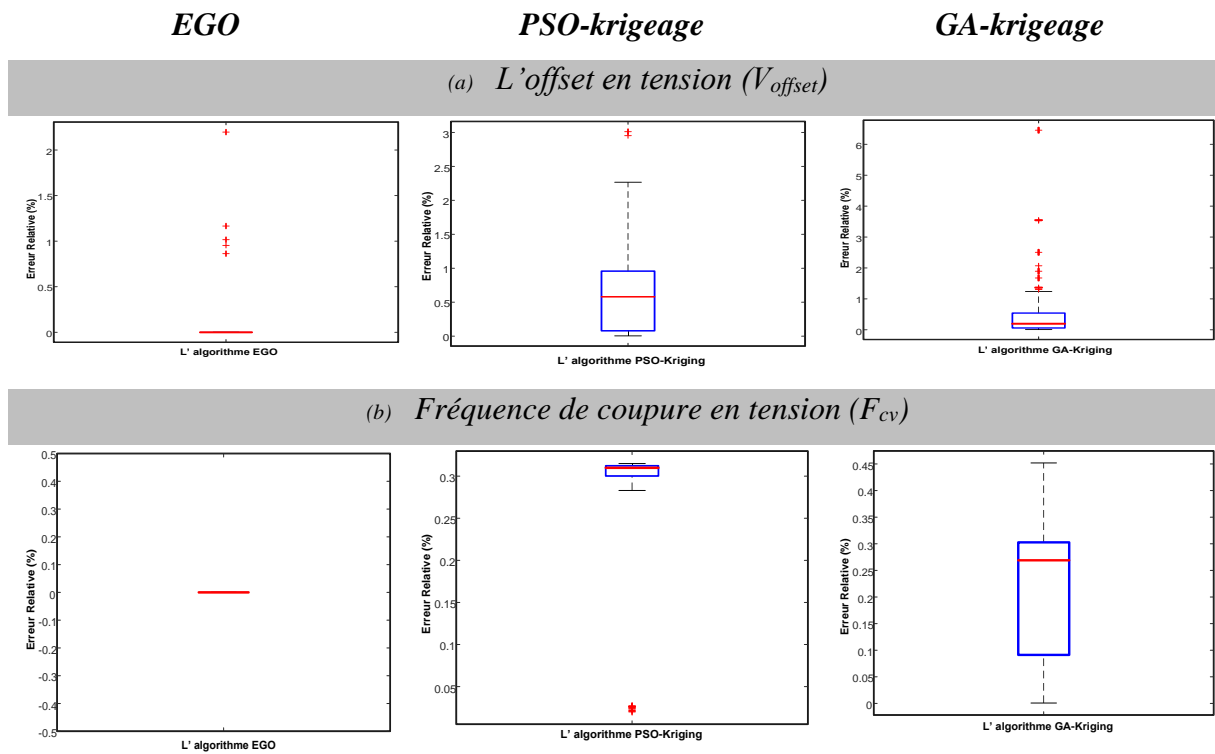


Figure 3.8 : Tests de la robustesse des erreurs relatives pour 100 itérations du circuit VF

D'après ces figures, nous avons conclu que l'algorithme EGO est plus robuste et plus précis avec une base de données initiale réduite et une erreur relative nulle.

3.3. Le critère d'amélioration '*Pseudo Expected Improvement*' (PEI)

3.3.1. Principe de fonctionnement de l'algorithme EGO parallèle

L'algorithme EGO a enrichi progressivement le plan d'expériences à chaque itération à travers le critère d'amélioration espérée « *Expected Improvement* » (EI). À la première itération, l'algorithme EGO choisit le point avec la valeur EI la plus élevée pour mettre à jour le modèle. Le point suivant est sélectionné à la deuxième itération en fonction du critère EI actualisé. Néanmoins, l'algorithme EGO gère le deuxième échantillon après avoir évalué le premier. Donc, cette approche ne peut évaluer les points que d'une manière séquentielle, et non pas d'une manière parallèle. Par conséquent, cette approche est plus coûteuse en temps.

Dans cette partie, nous proposons un nouveau critère nommé « *Pseudo expected improvement* » (PEI) pour développer un algorithme EGO parallèle (produire plusieurs points pour le calcul parallèle) [Zhan *et al.*, 2017] [Drira *et al.*, 2019]. Nous allons intégrer ce critère dans l'algorithme EGO pour sélectionner plusieurs points de conception à chaque itération. Ensuite, ces points peuvent être évalués en parallèle. Le premier point est sélectionné par la fonction initiale EI. Après, la fonction PEI est basée sur la multiplication de la fonction initiale de l'EI par une fonction d'influence (IF). De ce fait, ce critère PEI est capable de sélectionner de manière séquentielle plusieurs points par itération sans évaluer la fonction objectif. Ensuite, ces points peuvent être évalués en parallèle, ce qui permet de gagner en temps d'exécution.

Dans ce qui suit, nous détaillerons le fonctionnement de critère PEI dans l'algorithme EGO et nous présenterons une comparaison entre l'algorithme EGO conventionnel et l'algorithme EGO basé sur le critère PEI.

3.3.2. Le critère d'amélioration PEI

Le nouveau critère PEI a été intégré dans l'algorithme EGO pour sélectionner plusieurs points dans chaque itération. L'idée principale de l'approche proposée est de donner une approximation du critère EI en le multipliant par une fonction d'influence. De cette manière, dans une seule itération, plusieurs points sont obtenus et évalués simultanément.

a. La fonction d'influence (IF)

La valeur de la fonction IF est proportionnelle à la distance entre le point actuel et le point calculé (point d'actualisation). La fonction IF tend vers zéro lorsqu'elle est proche du point calculé et elle est égale à 1 lorsqu'elle est très loin du point calculé. Cette fonction est définie comme suit [Zhan *et al.*, 2017] :

$$IF(x, x^u) = 1 - Corr[\varepsilon(x), \varepsilon(x^u)] \quad (3.1)$$

où x^u est le point calculé de la fonction IF.

La fonction de corrélation est égale à 1 si les deux points sont vraiment proches l'un de l'autre et elle approche de zéro lorsque les deux points sont très éloignés. Par conséquent, si $x = x^u$, la valeur de la fonction IF est égale à zéro et elle est approchée de 1 en tant que la valeur de ($\|x - x^u\|$) tend vers l'infini.

b. Le critère pseudo expected improvement

Une nouvelle philosophie de sélection séquentielle de q points est évaluée relativement à la fonction IF. Supposons que n points initiaux $\{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$ sont utilisés pour construire le modèle de *krigeage*. La première fonction EI est calculée par l'équation (2.2) présentée dans le chapitre précédent et le premier point calculé est déterminé en maximisant la fonction EI initiale [Zhan *et al.*, 2017]:

$$x^{(n+1)} = \arg \max EI(x) \quad (3.2)$$

Ensuite, une approximation de la fonction EI est évaluée en multipliant la fonction initiale EI par la fonction IF de l'échantillon $x^{(n+1)}$:

$$PEI(x, x^{(n+1)}) = EI(x) \cdot IF(x, x^{(n+1)}) \quad (3.3)$$

Le deuxième point est créé en maximisant la nouvelle fonction EI :

$$x^{(n+2)} = \arg \max PEI(x, x^{(n+1)}) \quad (3.4)$$

Après, q-1 points sont ajoutés au plan d'expériences, la fonction PEI peut être dérivée comme suit :

$$PEI(x, x^{(n+1)}, x^{(n+2)}, \dots, x^{(n+q-1)}) = EI(x) \cdot IF(x, x^{(n+1)}) \cdot IF(x, x^{(n+2)}) \cdot IF(x, x^{(n+q-1)}) \quad (3.5)$$

Les q points sont définis comme suit :

$$x^{(n+i)} = \arg \max_{i \in [1, q]} PEI(x, x^{(n+1)}, x^{(n+2)}, \dots, x^{(n+i-1)}) \quad (3.6)$$

Ainsi, la fonction PEI est identifiée par :

$$PEI(x, q-1) = EI(x) \cdot IF(x, x^{(n+1)}) \cdot IF(x, x^{(n+2)}) \cdot IF(x, x^{(n+q-1)}) \quad (3.7)$$

où $q-1$ est le nombre de nouveaux points ajoutés au modèle.

En remplaçant (1.17) et (3.1) en (3.7), on obtient :

$$PEI(x, q-1) = EI(x) \cdot \prod_{i=1}^{q-1} \left[1 - \exp \left(- \sum_{k=1}^d \theta_k \cdot \left| x_k - x_k^{(n+i)} \right|^{P_k} \right) \right] \quad (3.8)$$

Le modèle de *krigeage* est construit. Il est à souligner que seuls les paramètres θ_k et P_k doivent être spécifiés dans la fonction PEI, ce qui facilite l'implémentation de ce dernier.

Dans ce qui suit, nous allons appliquer cette approche sur des fonctions tests et sur des circuits analogiques.

3.3.3. Application sur des fonctions tests et comparaison avec l'algorithme EGO conventionnel

Dans cette section, nous intégrons le critère (PEI) au lieu d'utiliser le critère (EI) dans l'algorithme EGO. Nous proposons une comparaison entre l'algorithme EGO-parallèle basé sur le nouveau critère PEI et l'algorithme EGO basé sur le critère EI. Ce travail montre une meilleure efficacité et rapidité de l'approche proposée par rapport à l'ancien algorithme EGO.

Au début, nous testons cet algorithme sur des fonctions tests qui sont présentées dans l'annexe afin de les comparer avec l'algorithme EGO conventionnel. Nous avons considéré les paramètres suivants : 10 points pour chaque itération pour l'algorithme EGO-PEI avec un maximum de 400 itérations pour les deux algorithmes et une population égale à 100 individus.

Le tableau 3.10 montre les résultats d'optimisation pour les deux algorithmes en mettant en relief le temps d'exécution de chaque approche.

D'après ces résultats, nous remarquons que les résultats d'optimisation des deux algorithmes sont très proches (on mesure une différence maximale de 0.57%) mais l'algorithme EGO-PEI présente une meilleure rapidité avec écarts moyens de 2 min. 2 sec.

Ensuite, nous avons effectué un test de robustesse pour trois fonctions de test pour vérifier et pour comparer les taux de convergence des deux algorithmes. Nous avons répété ces tests 100 fois pour chaque fonction et chaque algorithme. Puis, nous avons calculé le temps d'exécution moyen.

Tableau 3.10 : Performances des algorithmes : application sur des fonctions tests

	Résultats d'optimisation de l'algorithme EGO-EI	Résultats d'optimisation de l'algorithme EGO-PEI	Résultat théorique
F1			
<i>F(x)</i>	-1.0316	-1.0316	-1.0316
<i>X₁ et X₂</i>	X ₁ =-0.0899 X ₂ =0.7117	X ₁ =-0.0897 X ₂ =0.7118	
<i>Temps d'exécution</i>	5 min. 24 sec.	4 min.	
F3			
<i>F(x)</i>	0.3979	0.3979	0.3979
<i>X₁ et X₂</i>	X ₁ =9.4245 X ₂ =2.4738	X ₁ = 3.1416 X ₂ =2.2750	
<i>Temps d'exécution</i>	4 min. 54 sec.	3 min. 55 sec.	
F4			
<i>F(x)</i>	-3.862	-3.862	-3.8627
<i>X₁ et X₂</i>	X ₁ =0.1349 X ₂ =0.5583 X ₃ = 0.8545	X ₁ =0.1213 X ₂ =0.5539 X ₃ =0.8385	
<i>Temps d'exécution</i>	8 min. 39 sec.	7 min. 4 sec.	
F2I			
<i>F(x)</i>	-3.0393	-3.0336	-3.3223
<i>X₁ et X₂</i>	X ₁ =0.1934 X ₂ =0.1540 X ₃ = 0.4923 X ₄ =0.2723 X ₅ =0.3084 X ₆ =0.6481	X ₁ =0.1747 X ₂ =0.1610 X ₃ =0.4809 X ₄ =0.2682 X ₅ =0.3052 X ₆ =0.6458	
<i>Temps d'exécution</i>	21 min. 19 sec.	17 min. 9 sec.	

Le tableau 3.11 montre une présentation sous forme de boxplots des résultats des tests de robustesse pour les deux algorithmes et le temps moyen.

D'après ces figures, nous remarquons que les deux algorithmes sont robustes, et que l'algorithme EGO-PEI converge plus rapidement vers l'optimum en le comparant avec l'algorithme EGO-EI.

3.3.4. Application à des circuits analogiques

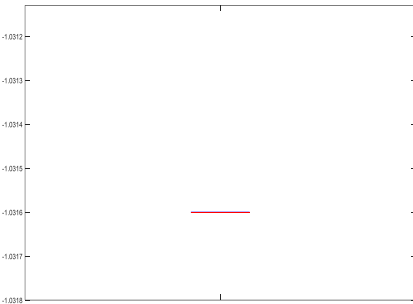
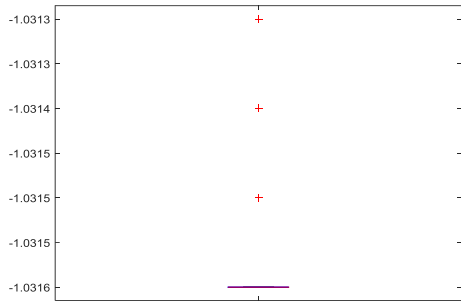
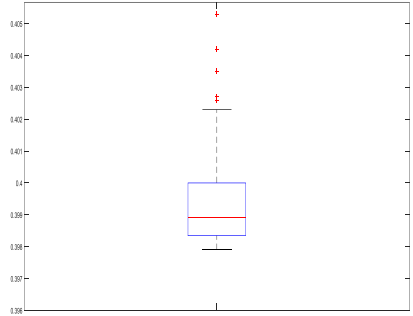
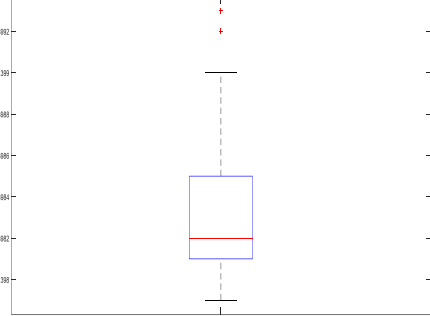
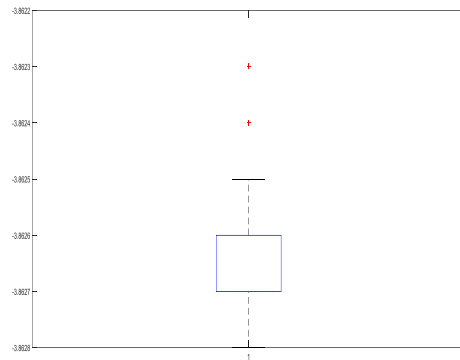
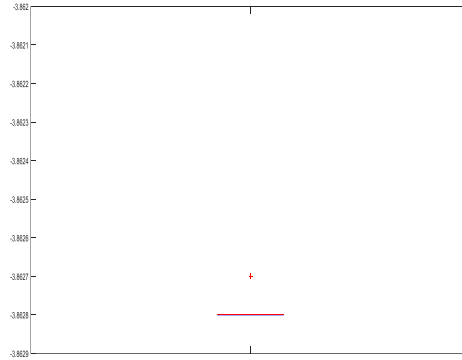
Dans cette section, nous présentons les résultats des tests des deux algorithmes (l'algorithme EGO conventionnel et l'algorithme EGO-PEI) sur deux circuits analogiques, à savoir, le circuit convoyeur de courant (CCII) et le circuit suiveur de tension (VF) qui sont présentés dans les sections précédentes.

3.3.4.1. Cas du convoyeur de courant

Dans ce qui suit, nous présenterons les résultats d'optimisation des deux algorithmes avec les performances du circuit convoyeur de courant CCII qui est présenté dans la figure 3.1. Les performances à optimiser du circuit CCII sont la résistance parasite au port X (R_x) et la

fréquence de coupure en courant (F_{ci}). Cette comparaison est effectuée avec une population égale à 100 individus pour les deux algorithmes et un maximum de 400 itérations.

Tableau 3.11 : Résultats des tests de robustesse pour les deux algorithmes

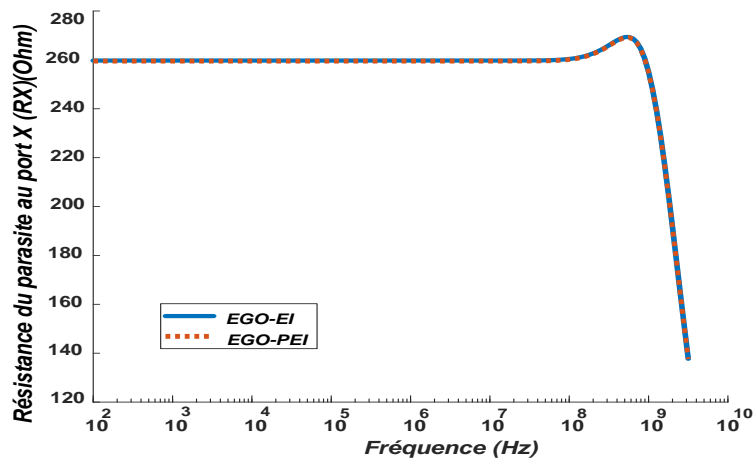
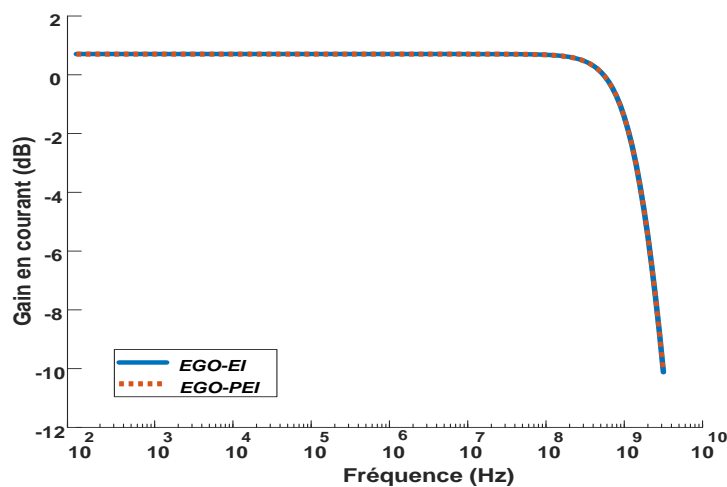
	<i>EGO-PEI</i>	<i>EGO-EI</i>
<i>F1</i>		
<i>Temps Moyen</i>	3 min. 47 sec.	4 min. 24 sec.
<i>F3</i>		
<i>Temps Moyen</i>	3 min. 56 sec.	4 min. 26 sec.
<i>F4</i>		
<i>Temps Moyen</i>	6 min. 29 sec.	7 min. 47 sec.

Le tableau 3.12 montre les résultats d'optimisation que nous avons obtenus et les résultats de simulation par l'outil HSPICE MT pour les deux algorithmes.

Tableau 3.12 : Les résultats de la simulation du circuit CCII

	Résultats optimisation d'algorithmes proposés	Résultats Simulation H-SPICE	Erreur Relative (%)	Paramètres (w_1, w_2) (μm)	Temps d'exécution
R_x					
<i>EGO-PEI</i>	259.78 Ω	259.53 Ω	0.09	$W_1=48.12$ $W_2=80$	3 min.7 sec.
<i>EGO-EI</i>	259.97 Ω	259.73 Ω	0.09	$W_1=47.98$ $W_2=80$	5 min.2 sec.
F_{ci}					
<i>EGO-PEI</i>	1.2350 GHz	1.226 GHz	0.76	$W_1=7.148$ $W_2=12.19$	3 min.19 sec.
<i>EGO-EI</i>	1.2347 GHz	1.226 GHz	0.75	$W_1=7.274$ $W_2=12.08$	6 min.10 sec.

Les figures 3.9 et 3.10 présentent les courbes de la simulation des deux performances, à savoir, la résistance parasite au port X (R_x) et le gain en courant (G_{ci}) avec les deux algorithmes EGO-PEI et EGO-EI.


Figure 3.9 : Courbe de la résistance du parasite au port X (R_x) du circuit CCII

Figure 3.10 : Courbe de gain en courant du circuit CCII

2.3.4.2. Cas du suiveur de tension

Dans ce qui suit, nous montrerons les résultats d'optimisation des deux algorithmes EGO-PEI et EGO-EI avec le circuit suiveur de tension qui est illustré dans la figure 3.4. Les deux performances à optimiser du circuit VF sont la tension d'offset (V_{Offset}) et la fréquence de coupure en tension (F_{cv}). Nous testons ces deux algorithmes avec une population égale à 100 individus et un maximum de 400 itérations.

Le tableau 3.13 présente les résultats d'optimisation et les résultats de simulation par l'outil HSPICE MT pour les deux algorithmes.

Tableau 3.13 : Les résultats de la simulation du circuit VF

	Résultats optimisation d'algorithmes proposés	Résultats Simulation H-SPICE	Erreur Relative (%)	Paramètres (w_1, w_2) (μm)	Temps d'exécution
V_{offset}					
<i>EGO-PEI</i>	-0.0995 V	-0.09898 V	0.55	$W_1=55.63$ $W_2=7.663$	3 min. 6 sec.
<i>EGO-EI</i>	-0.0995 V	-0.09899 V	0.54	$W_1=55.66$ $W_2=7.646$	6 min. 37 sec.
F_{cv}					
<i>EGO-PEI</i>	46.3996 MHz	46.406 MHz	0.0137	$W_1=48.97$ $W_2=48.38$	3 min. 19 sec.
<i>EGO-EI</i>	46,3997 MHz	46.406 MHz	0.0135	$W_1=48.71$ $W_2=48.40$	5 min. 15 sec.

Les figures 3.11 et 3.12 présentent les courbes de la simulation des deux performances, à savoir, la tension d'offset (V_{Offset}) et le gain en tension (G_{cv}).

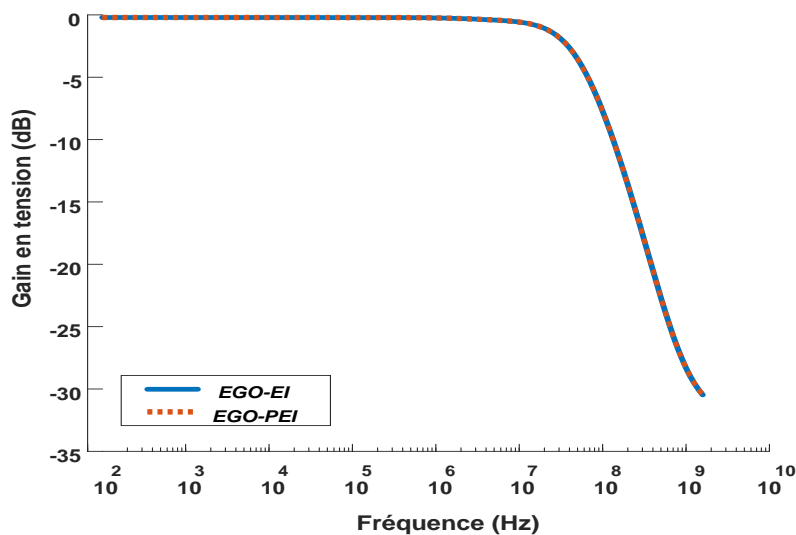


Figure 3.11 : Courbe de gain en tension du circuit VF

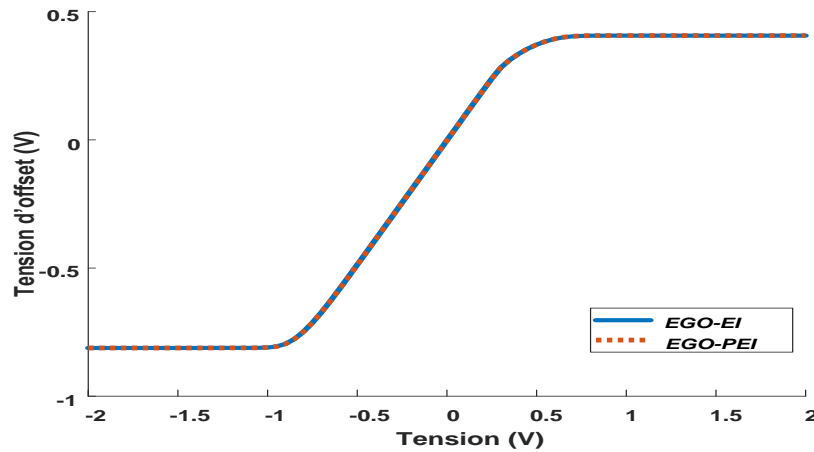


Figure 3.12 : Courbe de l'offset en tension (V_{offset}) du circuit VF

3.3.5. Récapitulatif et interprétations

D'après les résultats obtenus pour les fonctions tests et les deux circuits analogiques, nous constatons que les résultats de l'optimisation sont presque identiques pour les deux algorithmes (on mesure une différence maximale de 0.19 entre les deux algorithmes EGO-PEI et EGO-EI).

Nous remarquons aussi que les erreurs relatives sont faibles (la valeur maximum d'erreur relative est de 0.76%). Cependant, l'algorithme EGO-PEI est plus rapide que l'algorithme EGO conventionnel.

3.4. Adaptation de l'algorithme EGO pour l'optimisation des performances multiobjectif

Dans les tests précédents, nous avons montré l'efficacité et la rapidité de l'algorithme EGO mono-objectif pour la conception optimale des circuits analogiques [Drira *et al.*, 2018] [Drira *et al.*, 2019].

Dans ce qui suit, nous proposons l'utilisation de cette approche pour des problèmes multiobjectif. Dans la littérature il existe de nombreuses études qui ont été proposées pour généraliser l'algorithme EGO mono-objectif à l'optimisation multiobjectif [Jeong *et al.*, 2005] [Knowles, 2006] [Zhang *et al.*, 2009] [Zhan *et al.*, 2017].

L'algorithme EGO multiobjectif est une extension de l'algorithme EGO mono-objectif avec l'intention de gérer plusieurs objectifs. Cette méthode d'optimisation multiobjectif permet d'optimiser tous les critères d'amélioration espérée (EIs) de toutes les fonctions objectifs. Les solutions de Pareto obtenues par cet algorithme ont été considérées en tant que candidats prometteurs. Ces critères multiobjectif (EIs) ont la même routine que les algorithmes EGO mono-objectif [Zhan *et al.*, 2017]. Toutefois, lorsque le nombre de fonction objectifs est supérieur à deux, le critère EI en multiobjectif devient difficile à calculer, ce qui rend les calculs

de ces derniers généralement très coûteux [Couckuyt *et al.*, 2014] [Hupkens *et al.*, 2015]. Donc, un nouveau critère « *Expected Improvement Matrix* » (EIM) a été proposé [Zhan *et al.*, 2017] afin de réduire le temps de calcul.

3.4.1. Principe de fonctionnement de l'algorithme EGO multiobjectif

Dans ce qui suit, nous proposons une nouvelle approche MOEGO qui est développée en se basant sur le critère « *Expected Improvement Matrix* » (EIM), son but est de réduire le temps de calcul des critères (EI) en multiobjectif [Zhan *et al.*, 2017].

L'approche proposée construit une matrice pour le critère *Expected Improvement* (EI) qui contient toutes les informations des critères EIs à objectif unique sur tous les points non-dominés du front.

Le pseudo-code de l'algorithme MOEGO est présenté dans l'algorithme 3.1.

Algorithme 3.1: Algorithme MOEGO

Créer un plan d'expérience initial : $X = [x_1 ; \dots ; x_n]$
Évaluer la fonction en X et $Y_m = f_m(X)$ (avec m le nombre de fonctions)
Calculer f_{min} le minimum des résultats d'évaluations effectuées
Mettre à jour les solutions non-dominées
Tant que le critère d'arrêt n'est pas satisfait **Faire**
 Pour $i = 1$ to m
 Construire un modèle de Kriging pour chaque fonction objectif $(X ; Y)_m$
 fin Pour
 Calculer le EIM
 $X_{n+1} \leftarrow$ maximiser EIM ($\max EIM(x)$) et ajouter x_{n+1} à X .
 Pour $i = 1$ to m
 Évaluer $(Y_{n+1})_i \leftarrow f(x_{n+1})_i$ et ajouter $(y_{n+1})_i$ à Y_i .
 fin Pour
 Mettre à jour les solutions non-dominées
 Ré-estimer les paramètres et mettre à jour le modèle de krigeage
Fin Tant que

3.4.2. Le critère d'amélioration EIM

Pour l'optimisation multiobjectif, la meilleure solution actuelle est une matrice à deux dimensions [Zhan *et al.*, 2017] :

$$\begin{bmatrix} f_1^1 & \cdots & f_m^1 \\ \vdots & \ddots & \vdots \\ f_1^k & \cdots & f_m^k \end{bmatrix} \quad (3.9)$$

La meilleure solution actuelle f_{\min} en optimisation mutiobjectif se développe en deux directions : le nombre de points de la meilleure solution actuelle augmente de 1 à k et la dimension de chaque point passe de 1 à m.

Toutefois, la fonction scalaire EI (x) qui est appliquée pour l'optimisation mono-objectif peut également être étendue dans une matrice à deux dimensions pour l'optimisation multiobjectif, en particulier, la matrice d'amélioration espérée (EIM) [Zhan *et al.*, 2017] qui peut être définie par :

$$\begin{bmatrix} EI_1^1(x) & \cdots & EI_m^1(x) \\ \vdots & \ddots & \vdots \\ EI_1^k(x) & \cdots & EI_m^k(x) \end{bmatrix} \quad (3.10)$$

et

$$EI_i^j(x) = \left(f_i^j - \hat{y}_i(x) \right) \cdot \Phi \left(\frac{f_i^j - \hat{y}_i(x)}{s_i(x)} \right) + s_i(x) \cdot \phi \left(\frac{f_i^j - \hat{y}_i(x)}{s_i(x)} \right) \quad (3.11)$$

où $i = 1, 2, \dots, m$ et $j = 1, 2, \dots, k$. L'élément $EI_i^j(x)$ dans EIM représente l'amélioration espérée du point d'étude x au-delà de $j^{ième}$ les points non dominés du front dans le $i^{ième}$ objectif.

En réalité, la matrice d'amélioration espérée (EIM) est une idée simple pour traiter des problèmes multiobjectif. Cependant, le critère EIM ne donne aucune information complète dans la mesure où le point étudié peut améliorer l'approximation du front de Pareto.

Dans ce travail, nous avons utilisé le critère EIM à distance euclidienne qui a été développé pour la combinaison des éléments de la matrice EI. Le principal avantage de ce critère EIM proposé est sa haute efficacité et la réduction importante du temps de calcul.

L'amélioration de la distance euclidienne a été définie par Keane [Keane, 2006] comme la distance euclidienne entre le vecteur objectif de x et la solution non-dominée la plus proche :

$$I_e(x) = \min_{j=1}^k \sqrt{\sum_{i=1}^m (f_i^j - y_i(x))^2} \quad (3.12)$$

Le vecteur objectif du point x $y_i(x)$ est considéré en tant que variables aléatoires gaussiennes de dimension m.

La distance euclidienne basée sur le critère de EIM peut être définie comme suit:

$$EIM_e(x) = \min_{j=1}^k \sqrt{\sum_{i=1}^m (EI_i^j(x))^2} \quad (3.13)$$

Dans ce qui suit, nous allons tester l'algorithme MOEGO qui est basé sur le critère EIM à distance euclidienne sur des fonctions tests.

3.4.3. Application sur des fonctions tests

Dans le but de présenter les performances de l'algorithme MOEGO, nous appliquons cet algorithme pour l'optimisation des fonctions mathématiques telles que ZDT qui sont présentées en annexe. Ces dernières sont testées avec deux fonctions objectifs et 6 variables. Les paramètres de ce test sont comme suit : une population initiale égale à 50 individus et un nombre d'itérations égal à 200.

Le figure 3.13 montre les résultats d'optimisation des trois fonctions de test.

Le tableau 3.14 présente le temps d'exécution des trois fonctions de test.

Tableau 3.14 : Temps d'exécution des trois fonctions

Function 'ZDT1'	Function 'ZDT2'	Function 'ZDT3'
Temps :1 min. 12 sec.	Temps :1 min. 18 sec.	Temps :1 min. 15 sec.

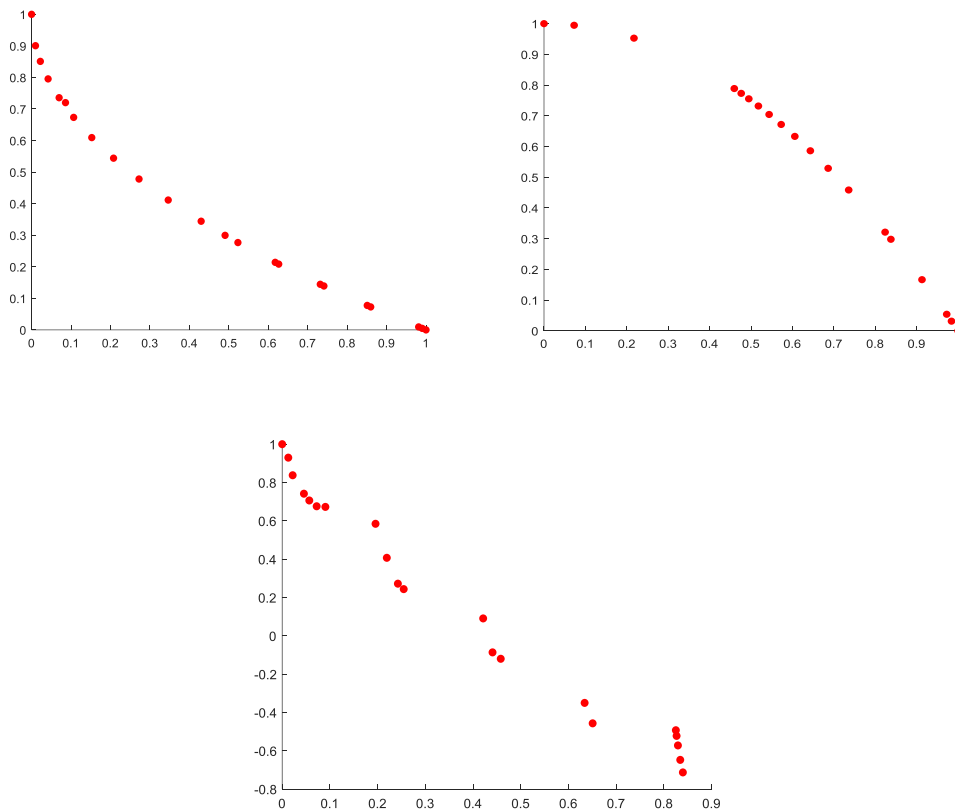


Figure 3.13 : Fronts de Pareto des trois fonctions

D'après les figures, nous remarquons que l'algorithme proposé permet de reproduire les mêmes fronts de Pareto (donnée par la littérature) avec un temps de calcul réduit.

Dans ce qui suit, nous présenterons les résultats d'optimisation obtenus en appliquant l'algorithme MOEGO présenté ci-dessus à des problèmes d'optimisation des performances d'un circuit analogique.

3.4.4. Application sur un circuit analogique

Dans cette section, nous présentons les résultats d'une comparaison entre l'algorithme MOEGO et l'algorithme MOPSO basé sur la simulation pour l'optimisation des performances du circuit convoyeur de courant qui est présenté dans la section 3.2.1. Le but de cette optimisation est de minimiser la résistance parasite au port X tout en maximisant la fréquence de coupure en courant. Deux variables (W_n , W_p correspondant aux largeurs des canaux des transistors NMOS et PMOS, respectivement) sont considérées. Nous avons considéré les paramètres suivants pour les deux algorithmes : une population initiale égale à 50 individus et un nombre d'itérations égal à 100.

La figure 3.14 montre les fronts de Pareto obtenus du circuit CCII par les deux algorithmes MOEGO et MOPSO.

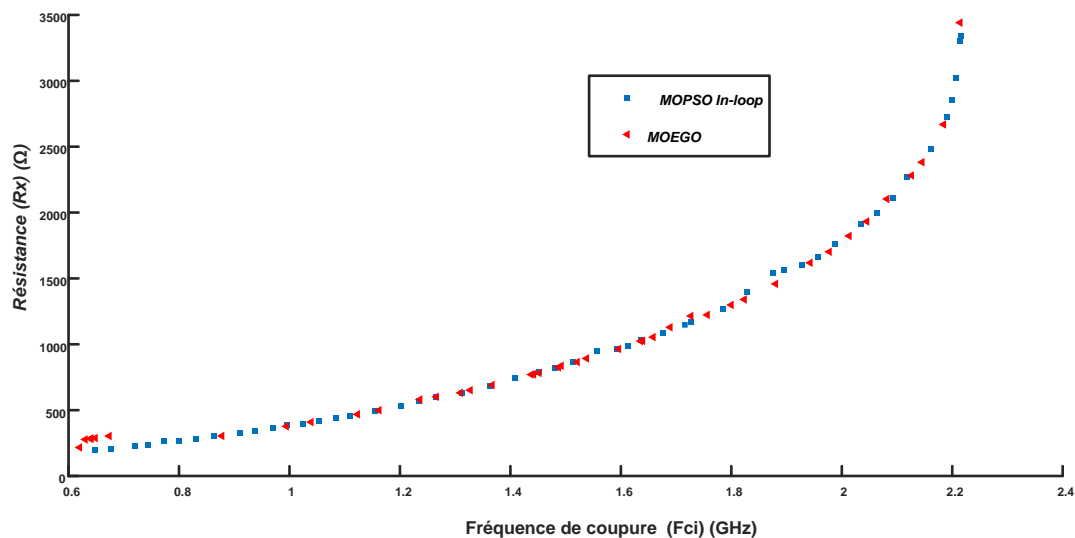


Figure 3.14 : Fronts de Pareto du circuit CCII

Le temps d'exécution de l'algorithme proposé est égal à 29 secondes pour générer le front, alors que le MOPSO basé sur la simulation nécessite 1h. 18 min pour obtenir « presque » le

même résultat, ce qui confirme l'efficacité de l'algorithme MOEGO, même avec un problème multiobjectif.

3.5. Conclusion

Dans ce chapitre, nous avons adapté et testé l'algorithme EGO pour la résolution des problèmes de dimensionnement des circuits analogiques. Au début, nous avons comparé cette approche avec deux métaheuristiques (PSO et GA) qui sont basées sur une technique de métamodélisation. Cette comparaison a été réalisée et analysée en utilisant une métrique « *Wilcoxon Signed Rank Test* ». Ces algorithmes ont été testés sur deux circuits analogiques, à savoir, le convoyeur de courant et le suiveur de tension. Ce travail montre la précision et la robustesse de l'algorithme EGO par rapport aux algorithmes basés sur le modèle de substitution, même avec un nombre limité d'échantillons initiaux.

Aussi, nous avons proposé une amélioration pour cette approche dans laquelle un nouveau critère, nommé *Pseudo expected improvement* (PEI) qui a été intégré dans l'algorithme EGO au lieu du critère *expected improvement* (EI). Une comparaison entre l'algorithme EGO basé sur le nouveau critère PEI et l'algorithme EGO conventionnel a été proposée pour montrer la rapidité et la précision de la nouvelle approche. Ce travail a donné lieu à une communication de conférence [Drira et al. 2019].

Nous avons aussi adapté cet algorithme pour l'optimisation des problèmes de dimensionnement multiobjectif. Nous avons développé l'algorithme EGO en multiobjectif en se basant sur le critère « *Expected Improvement Matrix* » (EIM) afin d'améliorer l'efficacité et de réduire le temps de calcul de cet algorithme. Cette approche a été testée sur des fonctions tests. Ensuite, elle a été appliquée à la conception du circuit analogique et elle a été aussi comparée à une métaheuristique basée sur la simulation MOPSO. Les résultats obtenus ont mis en relief les avantages de l'algorithme EGO en multiobjectif.

Conclusion générale et perspectives

Les résultats des travaux élaborés au cours de cette thèse se résument principalement à l'adaptation et l'amélioration de l'algorithme EGO à la conception et au dimensionnement optimal des circuits analogiques. Nous avons appliqué cette approche à des problèmes d'optimisation mono-objectif et multiobjectif afin de réduire le temps du cycle de conception de ces circuits.

L'algorithme EGO est basé sur l'utilisation de la technique de *krigeage*. Il permet d'ajouter les points supplémentaires d'amélioration à ce modèle pendant le processus d'optimisation à travers le critère d'amélioration espérée « *Expected Improvement* » (EI). L'étape d'optimisation est réalisée en appliquant l'algorithme de l'évolution différentielle (DE) afin d'effectuer une optimisation globale de la fonction EI dans l'algorithme EGO.

Dans ce travail, nous avons montré la précision et la rapidité de cette approche pour la résolution des circuits analogiques, même avec un faible nombre d'échantillons initiaux. Nous avons comparé cet algorithme avec des algorithmes métaheuristiques qui sont basés sur des outils de dimensionnement conventionnel tels que l'optimisation basée sur les simulations. Nous avons testé l'algorithme EGO et deux algorithmes (PSO, BSA) basés sur la simulation avec le circuit analogique « *Operational Transconductance Amplifiers* » (OTA). Comme résultat, on trouve que l'EGO présente une réduction considérable du temps de calcul par rapport aux autres algorithmes basés sur la simulation.

Une autre comparaison entre l'algorithme EGO et deux métaheuristiques (PSO et GA) basées sur la technique de *krigeage*, avec des fonctions de test et des circuits analogiques (le convoyeur de courant et le suiveur de tension) a été réalisée et analysée en utilisant la « *Wilcoxon Signed Rank Test* ». Nous avons montré que l'algorithme EGO est plus performant que ses concurrents en matière de précision et de rapidité.

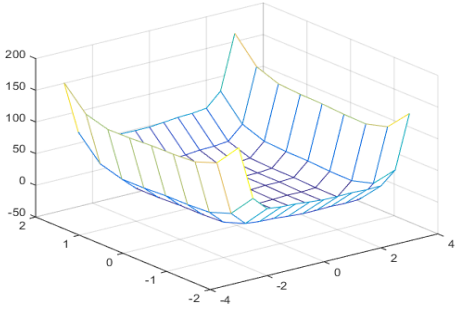
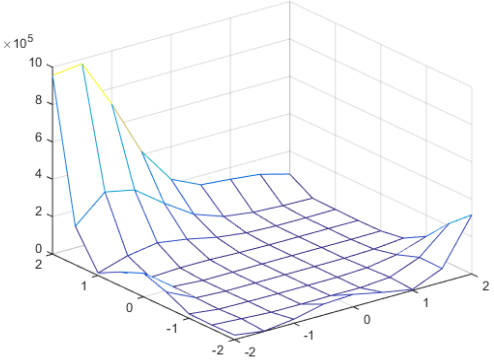
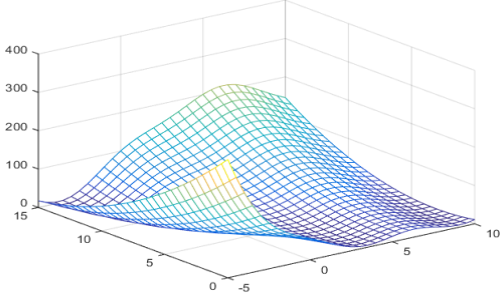
Aussi, nous avons apporté des améliorations au fonctionnement de cet algorithme. C'est ainsi que le critère d'amélioration espérée a été remplacé par le critère « *Pseudo expected improvement* » (PEI) qui a été proposé pour développer un algorithme EGO parallèle. Pareillement, nous avons adapté cet algorithme EGO mono-objectif à l'optimisation des problèmes multiobjectif en utilisant le critère « *Expected Improvement Matrix* » (EIM) afin de réduire le temps de calcul. Toutes les approches proposées ont été validées par des applications sur des circuits analogiques et comparées avec des performances obtenues par simulation. Ces résultats ont mis en relief les avantages des approches proposées.

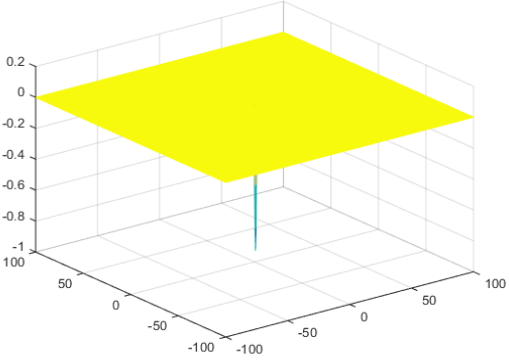
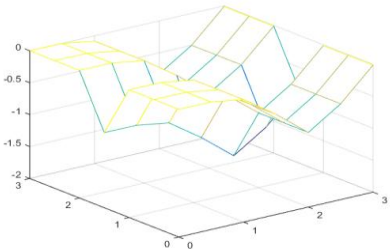
Au terme de ce travail de thèse nous avons projeté des perspectives qui se présentent comme suit :

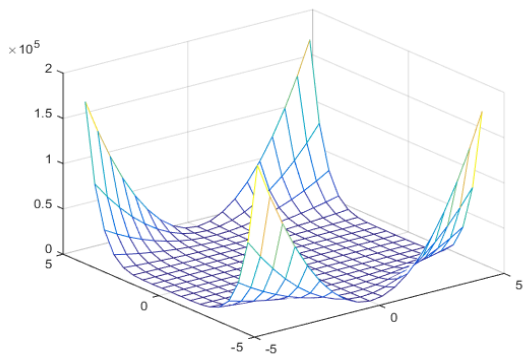
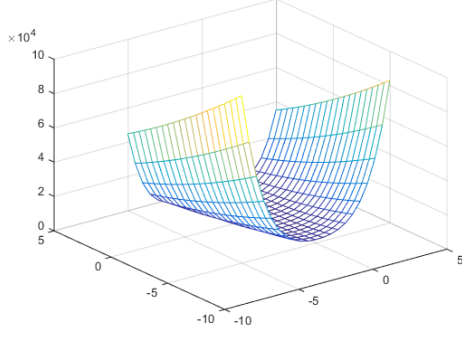
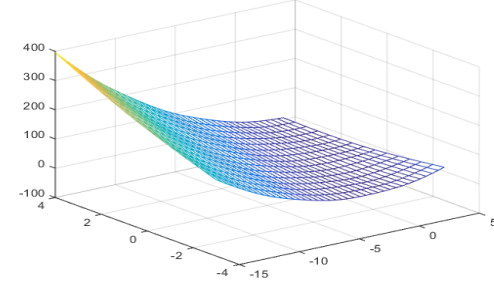
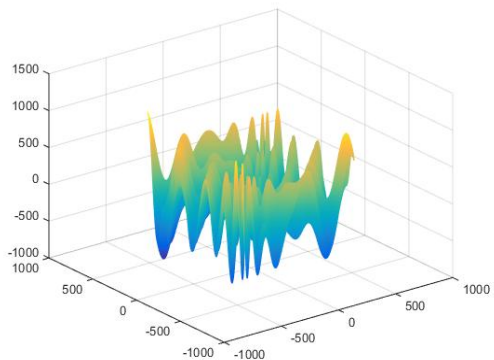
- Tester l'algorithme MOEGO pour la résolution d'autres circuits analogiques et utiliser des métriques de comparaison de fronts de Pareto.
- Intégrer un nouveau critère nommé « *Pseudo Expected Improvement Matrix* » (PEIM) dans l'algorithme MOEGO pour l'optimisation des problèmes multiobjectif rapides.

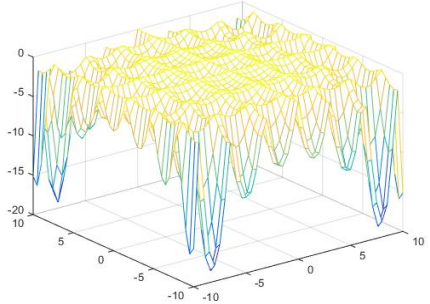
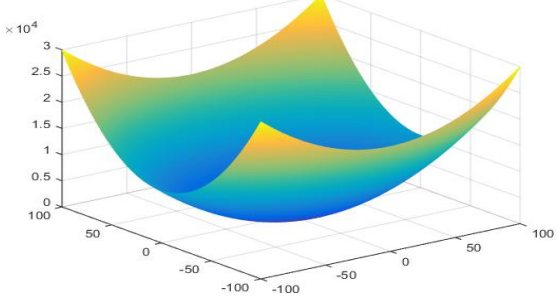
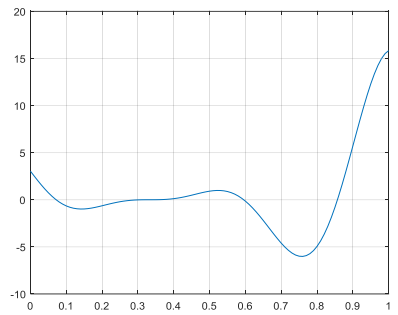
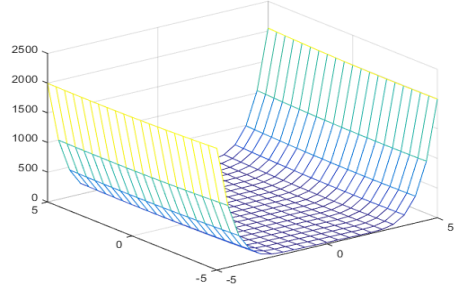
Annexe

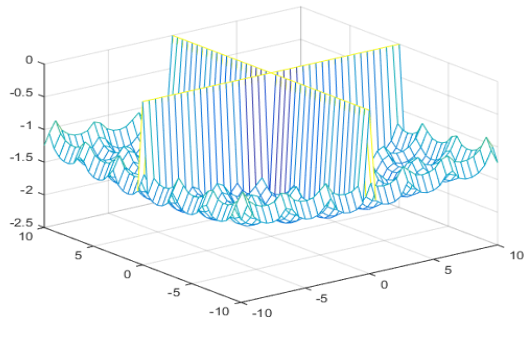
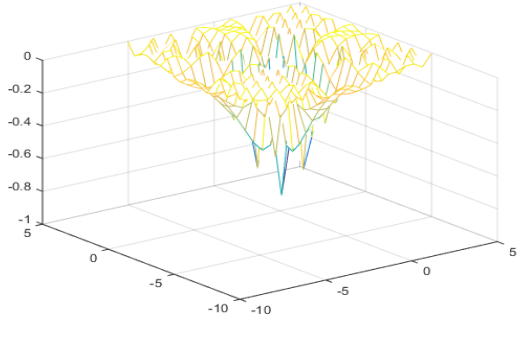
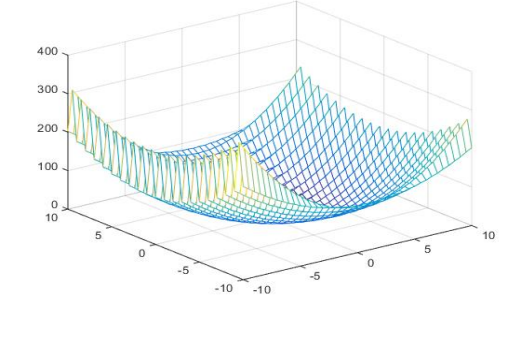

Les fonctions de test

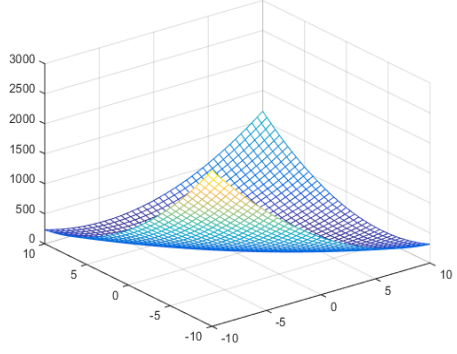
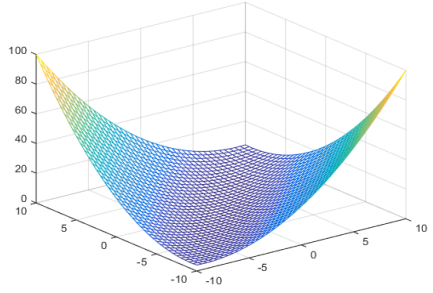
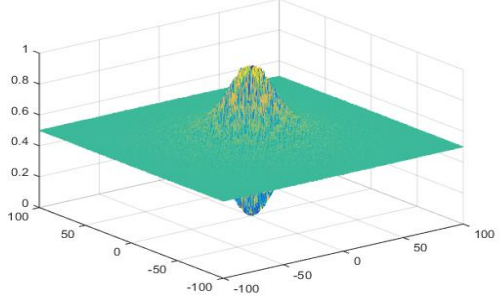
	<i>Fonctions</i>	<i>Figures de F(x)</i>
F1	<p><i>Six-hump Camel</i></p> $f(x) \left(4 - 2.1x_1^2 + \frac{x_1^4}{3} \right) x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$ $-3 \leq x_1 \leq 3$ $-2 \leq x_2 \leq 2$	
F2	<p><i>Goldstein-Price</i></p> $f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] * [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$ $-2 \leq x_1, x_2 \leq 2$	
F3	<p><i>Branin, or Branin-Hoo</i></p> $f(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t) \cos(x_1) + s$ <p>$a = 1, b = 5.1/(4\pi^2), c = 5/\pi, r = 6, s = 10$ and $t = 1/(8\pi)$</p> $-5 \leq x_1 \leq 10$ $0 \leq x_2 \leq 15$	

F4	<p>Hartmann 3-dimensional</p> $f(x) = - \sum_{i=1}^4 \alpha \exp\left(- \sum_{j=1}^3 A_{ij}(x_j - P_{ij})^2\right)$ $\alpha(1.0, 1.2, 3.0, 3.2)^T$ $A = \begin{pmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 30 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$ $P = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$ $0 \leq x_1, x_2, x_3 \leq 1$	
F5	<p>Easom</p> $f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$ $-100 \leq x_1, x_2 \leq 100$	
F6	<p>Michalewicz</p> $f(x) = - \sum_{i=1}^d \sin(x_i) \sin^{2m}(ix_i^2/\pi)$ $d = 2 ; m = 10 \quad 0 \leq x_1, x_2 \leq \pi$	

F7	<p><i>Beale</i></p> $f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$ $-4.5 \leq x_1, x_2 \leq 4.5$	
F8	<p><i>Rosenbrock</i></p> $f(x) = -100(x_2 - x_1^2)^2 + (x_1 - 1)^2$ $-5 \leq x_1, x_2 \leq 10$	
F9	<p><i>MCCORMICK</i></p> $f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$ $-1.5 \leq x_1 \leq 4$ $-3 \leq x_2 \leq 4$	
F10	<p><i>Eggholder</i></p> $f(x) = -(x_2 + 47) \sin\left(\sqrt{\left x_2 + \frac{x_1}{2} + 47\right }\right) - x_1 \sin(\sqrt{ x_1 - (x_2 + 47) })$ $-512 \leq x_1, x_2 \leq 512$	

<p>F11</p>	<p>HOLDER TABLE</p> $f(x) = -\left \sin(x_1) \cos(x_2) \exp\left(1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right) \right $ $-10 \leq x_1, x_2 \leq 10$	
<p>F12</p>	<p>BOHACHEVSKY</p> $f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$ $-100 \leq x_1, x_2 \leq 100$	
<p>F13</p>	<p><i>Forrester et al.'s</i></p> $f(x) = (6x - 2)^2 \sin(12x - 4)$ $0 \leq x \leq 1$	
<p>F14</p>	<p>Three-hump Camel</p> $f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$ $-5 \leq x_1, x_2 \leq 5$	

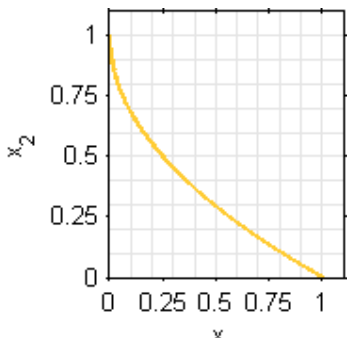
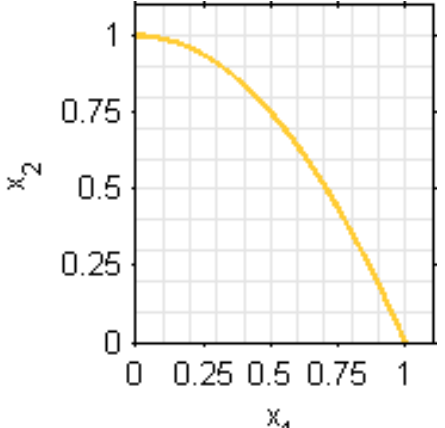
F15	<p><i>Cross-in-Tray</i></p> $f(x) = -0.0001 \left(\left s(x_1) \sin(x_2) \exp \left(\left 100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi} \right \right) + 1 \right \right)^{0.1}$ $-10 \leq x_1, x_2 \leq 10$	
F16	<p><i>Drop-Wave</i></p> $f(x) = -\frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2}$ $-5.12 \leq x_1, x_2 \leq 5.12$	
F17	<p><i>LEVY N. 13</i></p> <p>1.1. $f(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2 [1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2 [1 + \sin^2(2\pi x_2)]$</p> $-10 \leq x_1, x_2 \leq 10$	
F18	<p><i>BOOTH FUNCTION</i></p> $f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$ $-10 \leq x_1, x_2 \leq 10$	

		
F19	<p>MATYAS</p> $f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$ $-10 \leq x_1, x_2 \leq 10$	
F20	<p>Second Schaffer</p> $f(x) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$ $-100 \leq x_1, x_2 \leq 100$	
F21	<p>Hartmann 6-dimensional</p> $f(x) = - \sum_{i=1}^4 \alpha \exp\left(- \sum_{j=1}^6 A_{ij}(x_j - P_{ij})^2\right)$ $\alpha(1.0, 1.2, 3.0, 3.2)^T$	

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$$

$$P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

$$0 \leq x_1, x_2, x_3, x_4, x_5, x_6 \leq 1$$

<i>Fonctions</i>	<i>Pareto Front</i>
<p style="text-align: center;">ZDT 1</p> $f_1 = x_1$ $f_2 = g \cdot \left(1.0 - \sqrt{\frac{f_1}{g}} \right)$ $g(x_2, \dots, x_n) = 1.0 + \frac{9}{n-1} \sum_{i=2}^n x_i$ $0 \leq x_i \leq 1, i = 1, \dots, n$	<p style="text-align: center;">$x_2 = 1 - \sqrt{x_1}$</p> 
<p style="text-align: center;">ZDT 2</p> $f_1 = x_1$ $f_2 = g(\vec{x}) \cdot [1.0 - (x_1/g(\vec{x}))^2]$ $g(\vec{x}) = 1 + \frac{9}{n-1} \left(\sum_{i=2}^n x_i \right)$ $0 \leq x_i \leq 1, i = 1, \dots, n$	<p style="text-align: center;">$x_2 = 1 - x_1^2$</p> 

ZDT 3

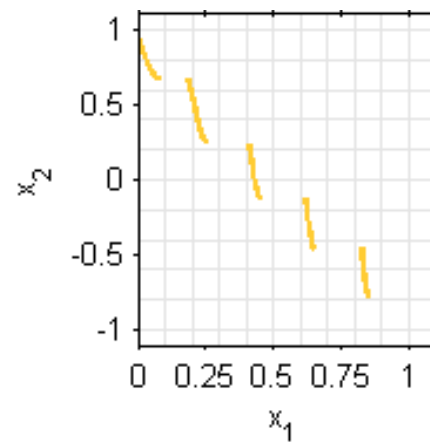
$$f_1(\vec{x}_1) = x_1$$
$$f_2 = g(\vec{x}) [1 - \sqrt{x_1/g(\vec{x})} - x_1/g(\vec{x}) \sin(10\pi x_1)]$$
$$g(\vec{x}) = 1 + \frac{9}{n-1} \left(\sum_{i=2}^n x_i \right)$$
$$0 \leq x_i \leq 1, i = 1, \dots, n$$

(x_1, x_2) avec $x_1 \in F$ et

$$x_2 = 1 - \sqrt{x_1} - x_1 \cdot \sin(10\pi x_1)$$

Où

$$F = [0, 0.0830015349] \cup$$
$$(0.1822287280, 0.2577623634] \cup$$
$$(0.4093136748, 0.4538821041] \cup$$
$$(0.6183967944, 0.6525117038] \cup$$
$$(0.8233317983, 0.8518328654]$$



Références bibliographiques

- [Aronhime *et al.*, 1978] P. Aronhime, M. Sharif. Bakhir, “A current conveyor realization using operational amplifier”, *International Journal of Electronics*, Vol. 45, pp. 283-288, 1978.
- [Barros *et al.*, 2010]. M. F. M. Barros, J. M. C. Guilherme, N. C. G. Horta, “Analog circuits and systems optimization based on evolutionary techniques”, Vol. 9, Springer, 2010.
- [Bennour *et al.*, 2010] S. Bennour, A. Sallem, M. Kotti, E. Gaddour, M. Fakhfakh, M. Loulou, “Application of the PSO technique to the Optimization of CMOS Operational Transconductance Amplifiers”, the International Conference on Design & Technology of Integrated Systems in Nanoscale Era, pp. 1-5, 2010.
- [Boyd *et al.*, 2001] S. Boyd, T. H. Lee, "Optimal design of a CMOS op-amp via geometric programming", *IEEE Transactions on Computer-aided design of integrated circuits and systems*, Vol. 20(1), pp. 1-21, 2001.
- [Bres *et al.*, 2006] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. “Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems”, *IEEE transactions on evolutionary computation*, Vol. 10(6), pp. 646-657, 2006.
- [Bres *et al.*, 2007] J. Brest, B. Boskovic, S. Greiner, V. Zumer, M. S. Maucec. “Performance comparison of self-adaptive and adaptive differential evolution algorithms”. *Soft Computing*, Vol. 11(7), pp. 617–629, 2007.
- [Brévilliers *et al.*, 2015] M. Brévilliers, O. Abdelkafi, L. Idoumghar, “Sequential and Parallel BSA Algorithm”, *Congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision (ROADEF)*, 2015.
- [Carr *et al.*, 1997] J. C. Carr, W. R. Fright, R. K. Beatson, “Surface interpolation with Radial Basis Functions for Medical Imaging”, *IEEE transactions on medical imaging*, Vol. 16(1), pp. 96-107, 1997.
- [Cavazzuti, 2012] M. Cavazzuti, “Optimization Methods: from Theory to Design Scientific and Technological Aspects in Mechanics”, Springer Science & Business Media, Berlin, 2012.
- [Civicioglu, 2013] P. Civicioglu, “Backtracking search optimization algorithm for numerical optimization problems”, Vol. 219(15), pp.8121-8144, 2013.

[Clerc *et al.*, 2004] M. Clerc, P. Siarry, “Une nouvelle métaheuristique pour l’optimisation difficile : la méthode des essais particuliers”, *J3eA*, Vol. 3, 2004.

[Coello *et al.*, 2002] C. A. Coello, M. S. Lechuga, “MOPSO: A proposal for multiple objective particle swarm optimization”, *The congress on evolutionary computation (CEC)*, Vol. 2, pp. 1051-1056. 2002.

[Conn *et al.*, 1996] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, C. Visweswariah, “Optimization of custom MOS circuits by transistor sizing”, *International conference on Computer-aided design*, pp. 174-180, 1996.

[Cooren, 2008] Y. Cooren, “Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire”, thèse de doctorat, Université Paris 12 Val de Marne, 2008.

[Couckuyt *et al.*, 2014] I. Couckuyt, D. Deschrijver, and T. Dhaene, “Fast calculation of multiobjective probability of improvement and expected improvement criteria for pareto optimization”, *Journal of Global Optimization*, Vol. 60(3), pp. 575-594, 2014.

[Daoud *et al.*, 2006] H. Daoud, S. B. Salem, S. Zouari, M. Loulou, “Folded cascode OTA design for wide band applications”, In *International Conference on Design and Test of Integrated Systems in Nanoscale Technology*, pp. 437-440, 2006.

[Darwin *et al.*, 1859] C. Darwin, “On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life”, H. Milford, Oxford University Press, 1859.

[Dastidar *et al.*, 2005] T. R. Dastidar, P. Chakrabarti, P. Ray, “Synthesis system for analog circuits based on evolutionary search and topological reuse”, *IEEE transactions on evolutionary computation*, Vol. 9(2), pp. 211-224, 2005.

[Deb, 2001] K. Deb, “Multi-objective optimization using evolutionary algorithms”, Wiley-Blackwell, Vol.16, 2001

[Deb *et al.*, 2002] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II”, *IEEE transactions on evolutionary computation*, Vol. 6(2), pp. 182-197, 2002.

[Degrauwe *et al.*, 1987] M. G. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B. Goffart, E. Vittoz, S. Cserveny, C. Meixenberger, G. Van Der Stappen, H. Oguey, “IDAC: an interactive

design tool for analog CMOS circuits”, IEEE journal of solid-State circuits, Vol. 22(6), pp. 1106-1116, 1987.

[Dorigo *et al.*, 2006] M. Dorigo, M. Birattari, T. Stützle, “Ant Colony Optimization : Artificial Ants as a Computational Intelligence Technique”, IEEE Computational Intelligence Magazine, Vol. 1(4), pp. 28-39, 2006.

[Dréo *et al.*, 2003] J. Dréo, A. Petrowski, P. Siarry, E. Taillard, “Métaheuristiques pour l’optimisation difficile”, Eyrolles, 2003.

[Drira *et al.*, 2018] N. Drira, M. Kotti, M. Fakhfakh, P. Siarry, E. Tlelo-Cuautle, “Expected Improvement-Based Optimization Approach for the Optimal Sizing of a CMOS Operational Transconductance Amplifier”, The IEEE International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), pp. 137-9, Czech Republic, 2018.

[Drira *et al.*, 2019] N. Drira, M. Kotti, M. Fakhfakh, P. Siarry, E. Tlelo-Cuautle, “Pseudo Expected Improvement Based Optimization for CMOS Analog Circuit Design”, The IEEE International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), pp. 137-140, Lausanne, Switzerland, 2019.

[Drira *et al.*, 2019] N. Drira, M. Kotti, M. Fakhfakh, P. Siarry, “Efficient Global ‘Rapid’ Optimization of Analog Circuits: Application to the Design of a CMOS Operational Transconductance Amplifier”, International Journal of Engineering Sciences and Research Technology (IJESRT), Vol. 8(9), pp. 105-114, 2019.

[El Dor, 2012] A. El Dor, “Perfectionnement des algorithmes d’optimisation par essaim particulière : applications en segmentation d’images et en électronique”, thèse de doctorat, Ecole doctorale mathématiques et STIC de l’université Paris-Est, France, 2012.

[El-Turky *et al.*, 1989] F. El-Turky, E. E. Perry, “BLADES: An artificial intelligence approach to analog circuit design”, IEEE transactions on computer-aided design of integrated circuits and systems, Vol. 8(6), pp. 680-692, 1989.

[Engrand, 1998] P. Engrand, “A multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management”, Electricité de France, 1998.

[Fakhfakh *et al.*, 2010] M. Fakhfakh, M. Loulou, “Live demonstration: CASCADES. 1: A flowgraph-based symbolic analyzer”, IEEE international symposium on circuits and systems (ISCAS), pp. 2782-2782, 2010.

[Fernandez *et al.*, 1996] F. V. Fernandez, A. Rodriguez-Vazquez, “Symbolic analysis tools - the state of the art”, The IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World (ISCAS), Vol. 4, pp. 798-801, 1996.

[Forrester *et al.*, 2008] A. Forrester, A. Sobester, A. Keane, “Engineering design via surrogate modelling: a practical guide”, John Wiley & Sons, 2008.

[Goldberg *et al.*, 1987] D. E. Goldberg, J. Richardson, “Genetic algorithms with sharing for multimodal function optimization”, Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, pp. 41-49, Lawrence Erlbaum, 1987.

[Goldberg, 1989] D. E. Goldberg, “Genetic Algorithms in Search, Optimization, and Machine Learning”, Studies in Computational Intelligence, 1989.

[Glover, 1989] F. Glover, “Tabu search-part 1”, ORSA Journal on Computing, Vol. 1(3), pp. 190-206, 1989.

[Gu, 2001] L. Gu, “A comparison of polynomial based regression models in vehicle safety analysis”, Design engineering technical conferences, ASME, 2001.

[Guerra-Gomez *et al.*, 2010] I. Guerra-Gomez, E. Tlelo-Cuautle, L. G. De la Fraga, “Sensitivity analysis in the optimal sizing of analog circuits by evolutionary algorithms”, The 7th international conference on electrical engineering, computing science and automatic control (CCE), pp. 381-385, Chiapas, México, 2010.

[Guerra, 2016] Guerra, “Optimisation multi-objectif sous incertitudes de phénomènes de thermique transitoire”, thèse de doctorat, Université de Toulouse, 2016.

[Harjani *et al.*, 1989] R. Harjani, R. A. Rutenbar, L. R. Carley, “OASYS: A framework for analog circuit synthesis”, IEEE transactions on computer-aided design of integrated circuits and systems, Vol. 8(12), pp. 1247-1266, 1989.

- [Hoare *et al.*, 2008] A. Hoare, D.G. Regan, D. P. Wilson, “Sampling and sensitivity analyses tools (SaSAT) for computational modelling”, *Theoretical Biology and Medical Modelling*, Vol. 5(1), pp. 4, 2008.
- [Holland, 1975] J. H. Holland, “Adaptation in natural and artificial systems”, Ann Arbor: University of Michigan Press, 1975.
- [Horn *et al.*, 1994] J Horn, N. Nafpliotis, D. E Goldberg, “A niched pareto genetic algorithm for multiobjective optimization”, *IEEE conference on evolutionary computation (CEC)*, Vol. 1, pp. 82-87, 1994.
- [Huelsman, 1996] L. P. Huelsman, “Symbolic analysis-a tool for teaching undergraduate circuit theory”, *IEEE transaction on education*, Vol. 39(2), pp. 243-250, 1996.
- [Hupkens *et al.*, 2015] I. Hupkens, A. Deutz, K. Yang, and M. Emmerich, “Faster exact algorithms for computing expected hypervolume improvement”, *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, pp. 65-79, 2015.
- [Jeong *et al.*, 2005] S. Jeong, S. Obayashi, “Efficient global optimization (EGO) for multi-objective problem and data mining”, *IEEE congress on evolutionary computation*, Vol. 3, pp. 2138-2145, 2005.
- [Jin *et al.*, 2001] R. Jin, W. Chen, T.W. Simpson, “Comparative studies of metamodeling techniques under multiple modelling criteria”, *Structural and Multidisciplinary Optimization*, Vol. 23(1), pp. 1-13, 2001.
- [Jones *et al.*, 1998] D.R. Jones, M. Schonlau, W. J. Welch, “Efficient global optimization of expensive black-box functions”, *Journal of Global Optimization*, Vol. 13, pp. 455-492, 1998.
- [Jourdan, 2005] A. Jourdan, “Planification d’expériences numériques”, *Revue Modulad*, Vol. 63(33), 2005.
- [Keane, 2006] A. J. Keane, “Statistical improvement criteria for use in multiobjective design optimization”, *The American Institute of Aeronautics and Astronautics (AIAA) Journal*, Vol. 44(4), pp. 879-891, 2006.
- [Karel *et al.*, 2011] C. Karel, L. Eric, D. Tom, “Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling”, *European Journal of Operational Research*, Vol. 214(3), pp. 683-696, 2011.

-
- [Kennedy *et al.*, 1995] J. Kennedy, R. C. Eberhart, “Particle swarm optimization”, IEEE International Conference on Neuronal networks, Vol. 4, pp. 1942-1948, 1995.
- [Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, “Optimization by simulated annealing”, Science, Vol. 220(4598), pp. 671-680, 1983.
- [Knowles, 2006] J. Knowles, “ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems”, IEEE Transactions on Evolutionary Computation, Vol. 10(1), pp. 50-66, 2006.
- [Koh *et al.*, 1990] H. Koh, C. Sequin, P. Gray, “OPASYN: a compiler for CMOS operational amplifier”, IEEE transaction on computer-aided design of integrated circuits and systems, Vol. 9(2), pp. 113-125, 1990.
- [Kotti, 2017] M. Kotti, “Contribution à l’automatisation de la conception des circuits RF : modélisation et optimisation des inductances planaires intégrées”, thèse de doctorat, Université de Sfax, 2017.
- [Mallipeddi *et al.*, 2015] R. Mallipeddi, M. Lee, “An Evolving Surrogate Model-Based Differential Evolution Algorithm”, Applied Soft Computing, Vol. 34, pp. 770-787, 2015.
- [Mancer, 2012] N. Mancer, “Contribution à l’optimisation de la puissance réactive en présence de dispositifs de compensation dynamique (FACTS)”, Master, Faculté des Sciences et de la technologie, 2012.
- [McKay *et al.*, 1979] M. D. McKay, R. J. Beckman, W. J. Conover, “A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code”, Technometrics, Vol. 21(2), pp. 239-245, 1979.
- [Medeiro *et al.*, 1994] F. Medeiro, R. Rodríguez-Macías, F. V. Fernández, R. Domínguez-Castro, J. L. Huertas, A. B. Rodríguez-Vázquez, “Global design of analog cells using statistical optimization techniques”, Analog integrated circuits and signal processing, Vol. 6(3), pp. 179-195, 1994.
- [Myers *et al.*, 2016] R. H. Myers, C. M. Douglas, C. M. Anderson-Cook, “Response surface methodology: process and product optimization using designed experiments”, John Wiley & Sons, 2016.

[Nie *et al.*, 1975] N. H. Nie, D. H. Bent, C. H. Hull, “SPSS: Statistical package for the social sciences”, Vol. 227, New York: McGraw-Hill, 1975.

[Pareto, 1964] V. Pareto, “Cours d'économie politique”, Librairie Droz, Vol. 1, 1964.

[Raquel *et al.*, 2005]. C. R. Raquel. and P. C. Naval, “An effective use of crowding distance in multiobjective particle swarm optimization”, Genetic and evolutionary computation conference, pp. 257-264, 2005.

[Sack *et al.*, 1989] J. Sack, W. J. Welch, T. J. Mitchell, H. P. Wynn, “Design and analysis of computer experiments”, Statistical science, Vol. 4(4), pp. 409-435, 1989.

[Sallem *et al.*, 2009] A. Sallem, E. Bradai, M. Kotti, E. Gaddour, M. Fakhfakh, M. Loulou, “Optimizing CMOS current conveyors through meta heuristics”, The IEEE International symposium on computational intelligence and intelligent informatics (ISCIII), pp. 167-171, 2009.

[Sallem, 2015] A. Sallem, “Développement d'une approche d'aide à la conception des circuits analogiques basée sur l'utilisation des métaheuristiques”, thèse de doctorat, Ecole Nationale d'Ingénieurs de Sfax (ENIS), Tunisie, 2015.

[Sallem *et al.*, 2010] A. Sallem., I. Guerra-Gómez, M Fakhfakh, M. Loulou, E. Tlelo-Cuautle, “Simulation-Based Optimization of CCII's Performances in Weak Inversion”, The IEEE International conference on electronics, circuits, and systems (ICECS), pp. 655-658 Gammarth, Tunisia, 2010.

[Sanchez-Sinencio, 2002] E. Sanchez-Sinencio, “Continuous-time filters from 0.1 Hz to 2.0 GHz”, In Proc. of XVII Conf. On Design of Circuits and Integrated Systems, Santander, Spain, 2002.

[Santner *et al.*, 2003] T. J. Santner, B. J. Williams, W. Notz, “The Design and Analysis of Computer Experiments”, Berlin, Springer, Vol. 1, 2003.

[Sawal Hamid, 2009] M. A. Sawal Hamid, “System level performance and yield optimization for analogue Integrated circuits”, thèse de doctorat, Université de Southampton, 2009.

[Schonlau, 1997] M. Schonlau, “Computer Experiments and Global Optimization”, thèse de doctorat, Université de Waterloo, 1997.

[Sedra *et al.*, 1990] A. S. Sedra, G. W. Roberts, F. Gohh, “The current conveyor: history, progress and new results”, IEEE proceedings G-Circuits, Devices and Systems, Vol. 137(2), pp. 78-87, 1990.

[Sedra *et al.*, 1970] A. Sedra, K. C. Smith, “A second generation current conveyor and its applications”, IEEE transactions on circuit theory, Vol. 17(1), pp. 132-134, 1970.

[Siarry *et al.*, 1997] P. Siarry, G. Berthiau, F. Durbin, J. Haussy, “Enhanced simulated annealing for globally minimizing functions of many-continuous Variables”, ACM transactions on mathematical software, Vol. 23(2), pp. 209-228, 1997.

[Siarry *et al.*, 2007] P. Siarry, Z. Michalewicz, “Advances in Metaheuristics for Hard Optimization”, Science & Business Media, Springer, 2007.

[Simpson *et al.*, 2001] W.S. Simpson, D. K. Lin, W. Chen, “Sampling Strategies for Computer Experiments: Design and Analysis”, International Journal of Reliability and Applications, Vol. 2(3), pp. 209-240, 2001.

[Simpson *et al.*, 1998] T. W. Simpson, T. M. Mauery, J. J. Korte, “Comparison of Response Surface and Kriging Models for Multidisciplinary Design Optimization”, Symposium on Multidisciplinary Analysis and Optimization, in AIAA, 1998.

[Simpson *et al.*, 2001] T. W. Simpson, D. J. L. Lin, W. Chen, “Sampling Strategies for Computer Experiments: Design and Analysis”, International Journal of Reliability and Applications, Vol. 2(3), pp. 209-240, 2001.

[Srinivas *et al.*, 1994] N. Srinivas, K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms”, Evolutionary computation journal, vol. 2(3), pp. 221-248, 1994.

[Storn *et al.*, 1997] R. Storn and K. Price. “Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces”, Journal of Global Optimization, Vol. 11(4), pp. 341-359, 1997.

[Surakamtron *et al.*, 1988] W. Surakamtron, P. Thitimajshima, “Integrable electronically tunable current conveyors”, IEEE proceedings G (*Electronic Circuits and Systems*), Vol. 135(2), pp.71-77, 1988.

[Tlelo-Cuautle, 2012] E. Tlelo-Cuautle, “Integrated Circuits for Analog Signal Processing”, Springer Science & Business Media, 2012.

[Todeschini, 2014] F. Todeschini, “Dimensionnement énergétique de réseaux de capteurs ultra-compactes autonomes en énergie”, thèse de doctorat, Université Paris-Saclay, 2014.

[Wheatley *et al.*, 1969] C. F. Wheatley, H. A. Wittlinger, “OTA obsolesces op. amp”, In P. Nat. Econ. Conf., pp. 152-157, 1969.

[Zhan *et al.*, 2017]. D. Zhan, J. Qian, Y. Cheng, “Balancing Global and Local Search in Parallel Efficient Global Optimization Algorithms”, Journal of Global Optimization, Vol. 67(4), pp. 873-892, 2017.

[Zhan *et al.*, 2017] D. Zhan, Y. Cheng, J. Liu, “Expected improvement matrix-based infill criteria for expensive multiobjective optimization”, IEEE Transactions on Evolutionary Computation, Vol. 21(6), pp. 956-975, 2017.

[Zhan *et al.*, 2017] D. Zhan, J. Qian, Y. Cheng, “Pseudo expected improvement criterion for parallel EGO algorithm”, Journal of Global Optimization, Vol. 68(3), pp. 641-662, 2017.

[Zhang *et al.*, 2009] Q. Zhang, W. Liu, E. Tsang, B. Virginas, “Expensive multiobjective optimization by MOEA/D with gaussian process model”, IEEE Transactions on Evolutionary Computation, Vol. 14(3), pp. 456-474, 2009.

[Zitzler, 1999] E. Zitzler, “Evolutionary algorithms for multiobjective optimization: Methods and applications”, thèse de doctorat, ETH Zurich, 1999.