Contribution à la synthèse et l'optimisation multi-objectif par essaims particulaires de lois de commande robuste RST de systèmes dynamiques

Riadh Madiouni

UNIVERSITÉ PARIS-EST

ÉCOLE DOCTORALE (ED 532)

MATHÉMATIQUES ET SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION (MSTIC)

THÈSE DE DOCTORAT

AVEC L'ÉCOLE NATIONALE D'INGÉNIEURS DE TUNIS (ENIT)

SPÉCIALITÉ INFORMATIQUE

par

Riadh MADIOUNI

Contribution à la synthèse et l'optimisation multi-objectif par essaims particulaires de lois de commande robuste RST de systèmes dynamiques

soutenue le 20 juin 2016 devant le Jury d'examen :

M. Faouzi	BOUANI	Professeur des Universités	Président
Mme. Lætitia	JOURDAN	Professeur des Universités	Rapporteur
M. Nacer K.	M'SIRDI	Professeur des Universités	Rapporteur
M. Soufiène	BOUALLÈGUE	Maître Assistant	Co-encadrant de Thèse
M. Joseph	HAGGÈGE	Maître de Conférences	Directeur de Thèse
M. Patrick	SIARRY	Professeur des Universités	Directeur de Thèse

À la mémoire de mon père Ali
À Hala, notre petite princesse

Remerciements

Je tiens tout d'abord à adresser mes plus vifs remerciements aux Directeurs de cette thèse, Messieurs Patrick Siarry, Joseph Haggège, et Mohamed Soufiène Bouallègue, pour leur direction avisée et exigeante, leurs qualités humaines et scientifiques et leur soutien constant tout au long de cette Thèse.

Je voudrais aussi exprimer ma gratitude à Madame Lætitia Jourdan, Professeur à l'Université Lille 1, à Monsieur M. Nacer K. M'Sirdi, Professeur l'Université Aix Marseille et Monsieur Faouzi Bouani, Professeur à l'ENIT et Directeur du Laboratoire LACS, qui ont bien voulu prendre de leur temps pour évaluer mon travail et donner leur avis. Leur jugement m'est très important.

Je souhaite également exprimer ma reconnaissance envers tous les membres du LiSSi et du LARA, pour toutes les conversations scientifiques ou non que l'on a pu avoir. Je remercie aussi Madame Sylvie Cach de l'ED MSTIC pour sa gentillesse et son aide.

Ce travail n'aurait pas pu être réalisé sans le soutien de ma famille, que je remercie tout particulièrement. Un grand merci à ma mère Hédia, à mon frère Mohamed et ma soeur Souhir, qui m'ont soutenu tout au long de mes études et dont je serai indéfiniment redevable.

Je tiens à présenter mes reconnaissances et mes remerciements à ma femme Hiba, qui n'a jamais cessé de me soutenir pour que je puisse finir ce travail doctoral et à qui je voudrais exprimer mes affections et mes gratitudes.

Résumé

Ces travaux de recherche portent sur la synthèse systématique et l'optimisation de correcteurs numériques à structure polynomiale RST par approches métaheuristiques. Les problèmes classiques de placement de pôles et de calibrage des fonctions de sensibilité de la boucle fermée RST sont formulés sous forme de problèmes d'optimisation multi-objectif sous contraintes pour lequel des algorithmes métaheuristiques de type NSGA-II, MODE, MOPSO et epsilon-MOPSO sont proposés et adaptés.

Deux formulations du problème de synthèse RST ont été proposées. La première approche, formulée dans le domaine temporel, consiste à minimiser des indices de performance, de type IAE et MO, issus de la théorie de la commande optimale et liés essentiellement à la réponse indicielle du système corrigé. Ces critères sont optimisés sous des contraintes non analytiques définis par des gabarits temporels sur la dynamique de la boucle fermée. Dans la deuxième approche de synthèse RST, une formulation dans le domaine fréquentiel est retenue. La stratégie proposée consiste à définir et calculer une fonction de sensibilité de sortie désirée en satisfaisant des contraintes de robustesse de type $\mathcal{H}_\infty$. L'utilisation des parties fixes dans la fonction de sensibilité de sortie désirée assurera un placement partiel des pôles de la boucle fermée RST. L'inverse d'une telle fonction de sensibilité désirée définira le filtre de pondération $\mathcal{H}_\infty$ associé.

Un intérêt particulier est porté à l'approche d'optimisation par essaim particulière PSO pour la résolution des problèmes multi-objectif de commande reformulés. Un algorithme MOPSO à grille adaptative est proposé et puis perfectionné à base des concepts de l'epsilon-dominance. L'algorithme epsilon-MOPSO obtenu a montré, par comparaison avec les algorithmes MOPSO, NSGA-II et MODE, des performances supérieures en termes de diversité des solutions de Pareto et de rapidité en temps de convergence. Des métriques de type distance générationnelle, taux d'erreurs et espacement sont toutefois considérées pour l'analyse statistique des résultats de mise en œuvre obtenus. Une application à la commande en vitesse variable d'un moteur électrique DC est effectuée, également pour la commande en position d'un système de transmission flexible à charges variables. La mise

en œuvre par simulations numériques sur les procédés considérés est également présentée dans le but de montrer la validité et l'efficacité de l'approche de commande optimale RST proposée.

Mots clés : Optimisation multi-objectif, optimalité de Pareto, essaims particulaires MOPSO, epsilon-dominance, algorithmes génétiques NSGA-II, évolution différentielle MODE, commande polynomiale RST, robustesse $\mathcal{H}_\infty$, performances nominales, gabarits fréquentiels, transmission flexible.

Abstract

This research focuses on the systematic synthesis and optimization of digital RST structure based controllers thanks to global metaheuristics approaches. The classic and hard problems of closed-loop poles placement and sensitivity functions shaping of RST control are well formulated as constrained multi-objective problems to be solved with proposed metaheuristics algorithms such as NSGA-II, MODE, MOPSO and especially epsilon-MOPSO.

Two formulations of the metaheuristics-tuned RST problem have been proposed. The first one, which is given in the time domain, deals with the minimization of several performance criteria like the Integral Absolute Error (IAE) and the Maximum Overshoot (MO) indices. These optimal criteria, related primarily to the step response of the controlled plant, are optimized under non-analytical constraints defined by temporal templates on the closed-loop dynamics. In the second approach, a formulation in the frequency domain is retained. The proposed strategy aims to optimize a desired output sensitivity function satisfying $\mathcal{H}_\infty$ robustness constraints. The use of a suitable fixed parts of the optimized output sensitivity function will provide partial poles placement of the closed-loop dynamics of the digital RST controller. The opposite of such desired sensitivity function will define the associated $\mathcal{H}_\infty$ weighting filter.

The Multi-Objective Particle Swarm Optimization (MOPSO) technique is particularly retained for the resolution of all formulated multi-objective RST control problems. An adaptive grid based MOPSO algorithm is firstly proposed and then improved based on the epsilon-dominance concepts. Such proposed epsilon-MOPSO algorithm, with a good diversity of the provided Pareto solutions and fast convergence time, showed a remarkable superiority compared to the standard MOPSO, NSGA-II and MODE algorithms. Performance metrics, such as generational distance, error rate and spacing, are presented for the statistical analysis of the achieved multi-optimization results. An application to the variable speed RST control of an electrical DC drive is performed, also for the RST position control of a flexible transmission plant with varying loads. Demonstrative simulations

and comparisons are carried out in order to show the validity and the effectiveness of the proposed metaheuristics-based tuning of the RST control approach, which is formulated in the multi-objective optimization framework.

Keywords : Multi-objective Optimization, Pareto optimality, Particle Swarm Optimization MOPSO, epsilon-dominance, Non-dominated Sorting Genetic Algorithm NSGA-II, Differential Evolution MODE, digital RST control, $\mathcal{H}_\infty$ robustness, nominal performances, frequency templates, flexible transmission system.

Table des matières

Remerciements **v**

Résumé **vii**

Abstract **ix**

Table des figures **xvi**

Liste des tableaux **xx**

Introduction générale **xxiii**

1 Contexte et état de l'art sur la synthèse et l'optimisation de la commande robuste **1**

1.1 Introduction 1

1.2 Généralités sur la commande des systèmes dynamiques 2

1.2.1 Structure des systèmes à commande numérique 2

1.2.2 Modélisation pour la commande 3

1.2.3 Synthèse de la commande 4

1.2.4 La robustesse et les incertitude 4

1.3 Commande robuste 4

1.3.1 Principe et concepts de base 4

1.3.2 Techniques de commande robuste 5

1.3.2.1 La commande $\mathcal{H}_\infty$ 5

1.3.2.2 La commande mixte $\mathcal{H}_2/\mathcal{H}_\infty$ 6

1.3.2.3 La commande adaptative 6

1.3.2.4 La commande prédictive 7

1.3.2.5 La commande multi-modèle 9

1.3.2.6 La commande CRONE 10
1.3.2.7 La commande polynômiale à structure RST 11
1.3.2.8 La commande par logique floue 11
1.3.2.9 La commande neuronale 12
1.3.2.10 La commande neuro-floue 14
1.4 Complexité de la synthèse et recours à l'optimisation 15
1.4.1 Problèmes complexes de synthèse de commande robuste 15
1.4.1.1 Synthèse de régulateurs RST 15
1.4.1.2 Synthèse et réglage des contrôleurs flous 15
1.4.1.3 Synthèse de la commande $\mathcal{H}_\infty$ 15
1.4.2 Solution proposée 16
1.4.3 Théorie de l'optimisation continue dynamique 16
1.4.3.1 Principe général 16
1.4.3.2 Classification des méthodes d'optimisation 19
1.4.3.3 Heuristiques et métaheuristiques 21
1.5 Métaheuristiques pour l'optimisation difficile 21
1.5.1 Généralités et concepts de base 21
1.5.2 Métaheuristiques à solution unique 22
1.5.2.1 Méthode GRASP 22
1.5.2.2 Recuit simulé 24
1.5.2.3 Méthode de recherche tabou 26
1.5.3 Métaheuristiques à population de solutions 26
1.5.3.1 Algorithmes d'intelligence en essaim 27
1.5.3.2 Algorithmes évolutionnaires 31
1.6 Optimisation multiobjectif de lois de commande robuste RST : position du problème 34
1.7 Conclusion 36

2 Algorithmes évolutionnaires d'optimisation multi-objectif 37
2.1 Introduction 37
2.2 Optimisation multi-objectif difficile 38
2.2.1 Notion de problème d'optimisation multi-objectif contraint 38
2.2.2 Notion de dominance 39
2.2.3 Approche agrégative 40

2.2.4 Approche non Pareto 41
2.2.5 Approche Pareto 41
2.3 Concepts de base des algorithmes évolutionnaires 43
2.4 Algorithmes génétiques multi-objectif 45
2.4.1 Principe de base d'un algorithme génétique mono-objectif 45
2.4.2 Paramètres de contrôle de l'algorithme 47
2.4.3 Techniques et processus d'évolution des générations 47
2.4.4 Pseudo code de l'algorithme 48
2.4.5 Algorithmes génétiques pour l'optimisation multi-objectif 48
2.4.5.1 L'algorithme NPGA 49
2.4.5.2 L'algorithme MOGA 52
2.4.5.3 L'algorithme MOMGA 53
2.4.6 Etude de l'algorithme NSGA-II 54
2.4.6.1 Principe et origines 54
2.4.6.2 Calcul de la distance de surpeuplement 56
2.4.6.3 Pseudo-code de l'algorithme NSGA-II 57
2.5 Algorithmes d'évolution différentielle multi-objectif 58
2.5.1 Aperçu historique 58
2.5.2 Principe et concept de base 60
2.5.2.1 Schémas de mutation 60
2.5.2.2 Opérateur de sélection 61
2.5.2.3 Pseudo-code de l'algorithme MODE 62
2.6 Métriques de performance en optimisation multi-objectif 64
2.6.1 Distance générationnelle 64
2.6.2 Espacement 64
2.6.3 Rapport d'erreur 65
2.6.4 Conclusion 65

3 Perfectionnement d'un algorithme d'optimisation par essaim particulaire multi-objectif 67
3.1 Introduction 67
3.2 L'optimisation par essaim particulaire 68
3.2.1 Formulation mathématique 68
3.2.2 L'algorithme PSO 69

3.2.3 Confinement des particules 70
3.2.4 Paramètres de l'algorithme PSO 72
3.2.4.1 Coefficients de confiance 72
3.2.4.2 Coefficient de constriction 72
3.2.4.3 Facteur d'inertie 73
3.2.5 Topologie du voisinage 74
3.2.6 Hybridations de l'algorithme PSO 75
3.3 Algorithme MOPSO proposé 76
3.3.1 Description de l'approche 76
3.3.2 Archive externe 79
3.3.3 Contrôleur d'archive 80
3.3.4 Grille adaptative 80
3.3.5 Pseudo-code de l'algorithme 81
3.4 Perfectionnement de l'algorithme MOPSO proposé 83
3.4.1 Méthode d'epsilon-dominance 83
3.4.2 Algorithme epsilon-MOPSO proposé 85
3.5 Mise en oeuvre et comparaison avec les techniques NSGA-II et MODE . . 86
3.6 Conclusion 95

4 Synthèse de régulateurs polynomiaux RST par l'algorithme epsilon-MOPSO 97
4.1 Introduction 97
4.2 Synthèse de régulateurs numériques RST 98
4.2.1 Structure RST de régulateurs numériques 98
4.2.2 Approche de synthèse par placement de pôles 100
4.2.3 Approche par placement de pôles et calibrage des fonctions de sensibilité 101
4.2.3.1 Principe de base 101
4.2.3.2 Fonctions de sensibilité 102
4.2.3.3 Calcul de la dynamique de régulation 103
4.2.3.4 Calcul de la dynamique de poursuite 103
4.3 Synthèse RST par la technique epsilon-MOPSO proposée 104
4.3.1 Critères de performance et contraintes pour la synthèse 104
4.3.1.1 Critères de performances temporels 104

4.3.1.2 Critères de performances fréquentiels 105
4.3.1.3 Contraintes de robustesse et de performance nominale . . 105
4.3.2 Formulation du problème d'optimisation 106
4.3.3 Mise en oeuvre pour la commande en vitesse d'un moteur DC . . . 107
4.3.3.1 Description du procédé 107
4.3.3.2 Résultats de simulation et discussion 107
4.4 Calibrage de fonctions de sensibilité par la technique epsilon-MOPSO . . . 113
4.4.1 Formulation du problème d'optimisation 113
4.4.2 Fonctions de sensibilité désirées 113
4.4.3 Critères de performance 115
4.4.4 Contraintes d'optimisation 116
4.4.5 Problème de synthèse RST multi-objectif formulé 116
4.4.6 Mise en oeuvre de l'approche de synthèse proposée 117
4.4.6.1 Description du système à commander 117
4.4.6.2 Résultats de simulation et discussion 118
4.5 Conclusion 125

Conclusion générale **127**

Annexe : Fonctions de test pour l'optimisation MO **131**

Bibliographie **133**

Table des figures

1.1 Schéma synoptique d'un processus commandé. 3
1.2 Forme standard de la synthèse $\mathcal{H}_\infty$. 5
1.3 Forme standard de la synthèse mixte $\mathcal{H}_2/\mathcal{H}_\infty$. 6
1.4 Principe de la commande adaptative directe. 8
1.5 Principe de la commande adaptative indirecte. 8
1.6 Principe de la commande prédictive GPC. 9
1.7 Principe de la commande multi-modèle. 10
1.8 Structure de la commande polynômiale RST. 11
1.9 Synoptique d'un régulateur par logique floue. 12
1.10 Exemple d'une structure de commande neuronale en phase d'apprentissage. 13
1.11 Exemple d'une structure de commande neuro-floue. 14
1.12 La fonction $f(x) = x^4 - 12x^3 + 47x^2 - 60x$, indiquant le minimum global x^* ainsi que le minimum local x_B^*. 18
1.13 Classification possible des différents types de méthodes d'optimisation [Collette & Siarry, 2004]. 20
1.14 Déplacement d'une particule dans l'essaim. 30
1.15 Faculté d'une colonie de fourmis à retrouver le plus court chemin, fortuitement obstrué par un obstacle [Tfaili, 2012]. 31
1.16 Principe d'un algorithme évolutionnaire standard [Coello.Coello et al., 2007]. 32

2.1 Représentation de l'espace de recherche et de son image par la fonction multi-objectif. 39
2.2 Illustration de la notion de dominance pour un problème d'optimisation bi-objectif. 40
2.3 Illustration de la méthode d'agrégation en cas d'optimisation bi-objectif : (a) Une frontière de Pareto convexe, (b) Une frontière de Pareto non convexe. A : domaine réalisable. 41

2.4 Front de Pareto dans le cas d'optimisation bi-objectif. 42
2.5 Composants clés de l'EA [Coello.Coello et al., 2007]. 43
2.6 Mutation au niveau d'un bit. 44
2.7 Croisement en un seul point. 44
2.8 Organigramme d'un AG standard. 46
2.9 Croisement en un point de deux chaînes. 47
2.10 Mutation de bit dans une chaîne. 47
2.11 Sélection d'individus avec le NPGA (d'après Horn et al. [Horn & Nafpliotis,1993]). 51
2.12 Principe de l'algorithme NSGA-II [Deb et al., 2002]. 56
2.13 Distance de crowding (surpeuplement) : cas d'un problème bi-objectif. . . . 57

3.1 Illustration du déplacement d'une particule dans l'espace de recherche en combinant les trois tendances du mouvement. 70
3.2 Différents types de topologie pour un essaim de particules : (a) en cercle, (b) en rayon, (c) en étoile. 75
3.3 Les cas possibles pour le contrôleur d'archive. 81
3.4 Représentation graphique de l'insertion d'une nouvelle solution dans la grille adaptative lorsque la particule se situe dans les frontières de la grille. 81
3.5 Représentation graphique de l'insertion d'une nouvelle solution dans la grille adaptative lorsque la particule se trouve en dehors des frontières de la grille. 82
3.6 Concept de la technique d'epsilon-dominance en optimisation multi-objectif. 84
3.7 Illustration du choix des solutions non dominées dans l'epsilon-Archive. . . 85
3.8 Fronts de Pareto produits pour la fonction de test F1. 87
3.9 Fronts de Pareto produits pour la fonction de test F2. 88
3.10 Fronts de Pareto produits pour la fonction de test F3. 88
3.11 Fronts de Pareto produits pour la fonction de test F4. 88
3.12 Front de Pareto produit pour la fonction de test F1. 92
3.13 Fronts de Pareto produits pour la fonction de test F5. 92
3.14 Fronts de Pareto produits pour la fonction de test F6. 93
3.15 Fronts de Pareto produits pour la fonction de test F7. 93

4.1 Boucle de régulation numérique avec un correcteur RST. 98
4.2 Formes générales des gabarits sur les fonctions de sensibilité [Zito, 2005]. . 102

4.3 Fronts de Pareto des algorithmes pour la résolution du problème (4.37). . . 109
4.4 Robustesse de l'algorithme epsilon-MOPSO vis-à-vis la variation paramétrique du facteur d'inertie. 110
4.5 Dynamique de convergence des algorithmes pour l'objectif $f_1(\boldsymbol{\theta})$ 110
4.6 Dynamique de convergence des algorithmes pour l'objectif $f_2(\boldsymbol{\theta})$ 111
4.7 Réponse fréquentielle de la fonction de sensibilité de sortie $S_{yp}(\boldsymbol{\theta}^*)$ 111
4.8 Réponse fréquentielle de la fonction de sensibilité d'entrée $S_{up}(\boldsymbol{\theta}^*)$ 112
4.9 Réponses indicielles du système en boucle fermée. 112
4.10 Diagramme schématique de la transmission flexible [Landau & Karimi, 1998].117
4.11 Les réponses fréquentielles des modèles en boucle ouverte. 119
4.12 Emplacement des pôles et des zéros du modèle sans charge. 120
4.13 Temps de calcul des algorithmes pour l'optimisation du correcteur RST : cas du premier modèle du système. 121
4.14 Fronts de Pareto des algorithmes pour le premier modèle du système. . . . 122
4.15 Réponse fréquentielle de la fonction de sensibilité de sortie du système. . . 123
4.16 Réponse fréquentielle de la fonction de sensibilité d'entrée du système. . . . 124
4.17 Réponses indicielles du système avec le correcteur RST optimisé. 124

Liste des tableaux

3.1 Temps de simulation en seconde des algorithmes proposés. 87
3.2 La métrique GD obtenu pour les fonctions de test. 89
3.3 La métrique SP obtenu pour les fonctions de test. 90
3.4 La métrique ER obtenu pour les fonctions de test. 91
3.5 Résultats pour la fonction F1 à deux objectifs. 93
3.6 Résultats pour la fonction F5 à deux objectifs. 94
3.7 Résultats pour la fonction F6 à trois objectifs. 94
3.8 Résultats pour la fonction F7 à trois objectifs. 94

4.1 Paramètres identifiés du moteur DC. 107
4.2 Résultats d'optimisation du problème (4.37), cas moyen. 108
4.3 Temps de calcul relatif aux algorithmes d'optimisation proposés. 108
4.4 Pôles et zéros du système de transmission flexible. 120
4.5 Coefficients optimisés de la fonction de sensibilité désirée RST. 121

Introduction générale

La commande robuste a attiré l'attention d'un nombre important d'automaticiens durant les dernières décennies. Plusieurs approches, développées dans ce cadre, ont trouvé de plus en plus d'applications dans la conduite de procédés industriels. Le problème de garantie de la robustesse dans le cas d'incertitudes paramétriques a suscité un grand intérêt de la part des ingénieurs. En effet, la prise en compte des imperfections de la modélisation et des perturbations externes est devenue un objectif incontournable en commande robuste. Parmi les problèmes rencontrés dans ce formalisme, de plus en plus complexes et délicats, figurent en premier lieu, ceux liés à la synthèse de correcteurs robustes à structure canonique RST. Ce type de correcteurs polynomiaux numériques est d'une utilisation très répandue pour la conduite de procédés industriels variés.

D'un autre côté, le recours aux métaheuristiques et à la théorie de l'optimisation difficile pour la résolution de problèmes génériques en automatique, tels que pour l'identification des modèles dynamiques, l'analyse et la synthèse de structures de commande robustes et performantes, a connu un essor particulier durant ces dernières décennies. La recherche d'une méthode systématique de synthèse permettant l'élaboration de lois de commande robuste, tout en réduisant au mieux le temps de conception et les ressources de calcul, constitue une préoccupation d'actualité dans la recherche en théorie de la commande robuste. Pour mieux gérer cette complexité, le recours à la théorie de l'optimisation présente une solution prometteuse, surtout avec les puissances de calcul et des logiciels de CAO à disposition. Les problèmes de synthèse, en particulier ceux relatifs à la commande numérique polynomiale RST, peuvent être reformulés sous forme de problèmes d'optimisation multicritères pour lesquels une multitude d'algorithmes métaheuristiques peuvent être adoptée.

Nos travaux de recherche rentrent ainsi dans ce cadre et tentent de développer des lois de commande robuste RST en se basant sur des approches métaheuristiques globales. Un intérêt particulier est porté à la technique d'optimisation par essaims particulaires PSO,

jugée plus robuste et plus performante que ses homologues. Une formulation dans le cadre de l'optimisation multi-objectif de cette technique est proposée. La même méthode est ensuite perfectionnée, selon les concepts de l'epsilon-dominance, dans le but d'améliorer les capacités de son algorithme en termes de rapidité de convergence et de diversité des solutions de Pareto obtenues. L'algorithme perfectionné epsilon-MOPSO ainsi implémenté présentera l'outil de base pour la résolution des différents problèmes de synthèse RST formulés. D'autres algorithmes métaheuristiques, en l'occurrence le NSGA-II, le MODE et le MOPSO, sont également introduits et mis en œuvre comme supports de comparaison.

Cette thèse a été préparée conjointement au sein de l'université de Paris-Est, dans le Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi, E.A.3956) et à l'Ecole Nationale d'Ingénieurs de Tunis (ENIT), dans le LAboratoire de Recherche en Automatique (LARA, LR11-ES18). Ce travail a été proposé et encadré par Monsieur Joseph Haggège, Maître de Conférences à l'ENIT et Monsieur Patrick Siarry, Professeur des universités et Directeur de l'équipe Traitement de l'Image et du Signal SIgnal, IMage et Optimisation (SIMO). Il a été co-encadré par Monsieur Mohamed Soufiene Bouallègue, Maître assistant à l'Institut Supérieur des Systèmes Industriels de Gabès.

Le plan de ce mémoire est organisé comme suit : Dans le premier chapitre, nous avons présenté, en premier lieu, des généralités sur la théorie de la commande robuste. Les principales approches de synthèse ont été évoquées dans le but de montrer la complexité de leur résolution par des méthodes classiques connues. Les problèmes de commande numérique à structure canonique RST de systèmes complexes ont été plus particulièrement soulignés. Nous avons clôturé ce chapitre par la présentation de quelques métaheuristiques de voisinage et d'autres à base de population, plus particulièrement la méthode des essaims particulaires PSO. Enfin, nous avons justifié le choix de recours à l'optimisation multi-objectif comme solution proposée vis-à-vis la problématique abordée dans les présents travaux.

Le deuxième chapitre est consacré, en premier lieu, à la présentation de quelques définitions de base et des notations formelles sur les algorithmes évolutionnaires en optimisation multi-objectif. Les définitions d'un problème d'optimisation multi-objectif sous contraintes, de la dominance des solutions ainsi que des concepts d'optimalité au sens de Pareto sont présentées. En second lieu, ce chapitre présente aussi une analyse des

algorithmes génétiques NSGA, et puis NSGA-II, ainsi que les algorithmes à évolution différentielle MODE. Quelques métriques de mesure de performance des algorithmes métaheuristiques en optimisation multi-objectif sont présentées en détails afin de pouvoir valider les algorithmes proposés.

L'objet du troisième chapitre est de présenter l'algorithme d'optimisation par essaims particulaires multi-objectif MOPSO qui sera ultérieurement l'outil de base pour la résolution des problèmes de synthèse RST reformulés. Le concept d'epsilon-dominance est introduit en vue d'améliorer les capacités de cette approche en termes de rapidité de convergence et de diversité des solutions fournies sur les surfaces de compromis. Dans ce chapitre, les différents concepts de base ainsi que le mode de fonctionnement de l'approche MOPSO sont soulignées. Enfin, une mise en œuvre des algorithmes MOPSO, epsilon-MOPSO, NSGA-II et MODE, ainsi que des comparaisons des résultats sont effectuées sur un benchmark de fonctions de test issues de la littérature de l'optimisation multi-objectif.

Le quatrième et dernier chapitre de cette thèse est dédié à l'application des techniques métaheuristiques proposées, en particulier l'algorithme perfectionné epsilon-MOPSO, pour la synthèse systématique et aisée de correcteurs numériques à structure polynomiale RST reformulée sous forme de problèmes d'optimisation multi-objectif. Deux nouvelles formulations de la synthèse RST ont été proposées dans ce cadre afin de surmonter les problèmes délicats de placement des pôles de la boucle fermée et de calibrage des fonctions de sensibilité. Nous avons commencé par une première formulation de synthèse dans le domaine temporel qui consiste à minimiser des objectifs de performance issus de la théorie de la commande optimale et liés essentiellement à la réponse indicielle du système corrigé comme les critères IAE (Integral Absolute Error) et MO (Maximum Overshoot). Ces critères, présentant des gabarits temporels, permettent d'imposer des contraintes non linéaires sur la dynamique de la boucle fermée. Dans la deuxième approche, nous avons opté pour une formulation dans le domaine fréquentiel. La stratégie proposée consiste à définir et calculer une fonction de sensibilité de sortie désirée en satisfaisant des contraintes de robustesse liées à la nome $\mathcal{H}_\infty$ des filtres de pondération relatifs. La formulation du problème d'optimisation est basée essentiellement sur l'exploitation de l'égalité de Zames-Francis, connue sous le critère intégral de Poisson. Des applications sur un moteur électrique de type DC et un procédé de transmission flexible à charges variables sont effectuées dans le but de montrer l'efficacité de l'approche proposée.

Enfin, dans la conclusion générale du manuscrit, nous récapitulons nos contributions et nous proposons des perspectives, sur la base des travaux effectués.

Chapitre 1

Contexte et état de l'art sur la synthèse et l'optimisation de la commande robuste

1.1 Introduction

La théorie de la commande robuste a atteint ces dernières décennies un degré de maturité remarquable. Beaucoup de techniques de commande sont disponibles, et nombreux sont les problèmes auxquels sont confrontés les ingénieurs pour commander des systèmes industriels [Borne et al., 1993a, 1993b].

La recherche d'un outil et d'une procédure qui seront exploitables du point de vue pratique est une exigence désormais. Les théories de synthèse existantes des procédures réalisant un système de contrôle ont une complexité remarquable avec un temps d'investissement très important. Face à cette complexité, le recours à l'optimisation est une méthode prometteuse et systématique pour la synthèse des lois de commande robuste. L'évolution et la puissance de la théorie de l'optimisation et l'utilisation de cette dernière pour plusieurs problèmes dans divers domaines justifient davantage son utilisation pour la synthèse de la commande robuste [Borne et al., 1992a, 1992b].

Ce premier chapitre s'efforce de présenter d'abord un aperçu général sur la commande robuste des systèmes dynamiques. Ensuite, nous présentons le principe, les concepts de base et quelques techniques de la commande robuste. Puis nous détaillons les problèmes de synthèse relatifs à quelques techniques de commande, ainsi que les diverses méthodes d'optimisation connues. Enfin, nous justifions nos orientations vers l'optimisation multi-

objectif qui constitue le sujet principal de ce travail de thèse.

1.2 Généralités sur la commande des systèmes dynamiques

On rappelle dans cette section quelques notions sur les systèmes dynamiques et leur commande avant d'aborder la problématique qui nous intéressera par la suite. Aussi, il sera très important de savoir et comprendre ce que l'on entend par "systèmes dynamiques", car ceux-ci constituent la modélisation mathématique et la définition d'une approche de commande [Borne et al., 1993a, 1993b ; Borne et al., 1992a, 1993b ; Bonnet, 2008 ; Ayadi, 2002].

La notion de système dynamique se rencontre dans tous les éléments du monde physique. Tous ces éléments sont en évolution continue au cours du temps, et chaque système dépend de l'ensemble des informations qui le caractérisent. L'objectif est de commander les systèmes, c'est-à-dire imposer un comportement convenable qui tient compte de tous les effets extérieurs [Borne et al., 1992a ; Larminat, 2007 ; Bühler, 1988 ; Thierry, 2000]. Etant donné l'importance de cette phase, la conception doit être fiable, en termes de rendement du système sur les plans de la performance et de la qualité. Pour les concepteurs, fournir la méthodologie pour la commande des systèmes industriels constitue la première tâche, vu que la mise en oeuvre des stratégies de commande touche plusieurs domaines, tels que l'électro-hydraulique, l'électromécanique, l'électrothermique et l'aéronautique.

1.2.1 Structure des systèmes à commande numérique

Lorsque l'on envisage la commande d'un système, la première étape consiste à le modéliser. Modéliser un système consiste à élaborer un objet mathématique qui permet de décrire et prédire son comportement dynamique lorsque ce dernier est soumis à des influences externes. Ceci consiste à élaborer un modèle mathématique permettant d'appliquer des méthodes pour améliorer le comportement dynamique du système envisagé [Green & Limebeer, 1995 ; Borne et al., 1992a, 1992b]. Cette phase nécessite la construction d'un modèle bien clair et précis, souvent complexe, pour la synthèse d'une loi de commande par des méthodes disponibles.

Comme le montre la figure 1.1, le système de commande d'un processus dynamique met en oeuvre une loi de commande à partir des informations disponibles sur le modèle,

les consignes, les mesures de sorties et des informations sur les perturbations et le bruit. Le comportement dynamique du système est celui de ses variables de sortie au cours du

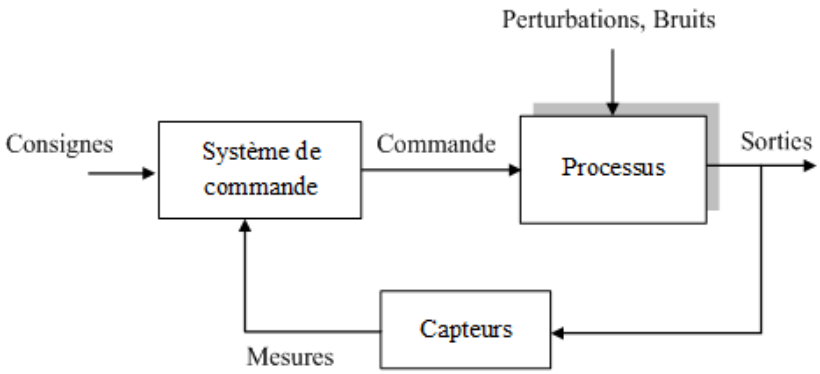

FIGURE 1.1 – Schéma synoptique d'un processus commandé.

temps, relativement à l'évolution des variables d'entrée. Ce comportement est à distinguer du comportement statique, où les variables de sortie reproduisent, à un gain près, le comportement des variables d'entrée. Suivant ce que l'on connaît du système, on peut distinguer deux types de modèles :

- le modèle de connaissance, où apparaissent les équations de fonctionnement du système, ainsi que ses variables internes ;
- le modèle de comportement, qui exprime les relations entre les entrées et les sorties du système.

Ainsi, commander un système revient à résoudre quelques problèmes relevant des sciences de l'ingénieur et portant sur les notions de modélisation, d'analyse et de synthèse de lois de commande développées [Zhou et al., 1996 ; Green & Limebeer, 1995 ; Chouiter, 1997, Zhengchao et al., 2015 ; Rotella & Fourquet, 2001].

1.2.2 Modélisation pour la commande

La théorie des systèmes est fondée sur un paradigme de modélisation. Les entrées-sorties captent respectivement l'influence de l'environnement sur le système et celle de ce dernier sur son environnement. Par conséquent, la modélisation est la transcription abstraite d'une réalité. Comme une première phase, la modélisation sert à donner le modèle analytique convenable du système en se basant sur des principes de la physique. Les paramètres du système sont généralement calculés en se basant sur des observations et des essais expérimentaux [Borne et al., 1992a, 1992b ; Caron & Hautier, 1995 ; Thierry, 2000].

1.2.3 Synthèse de la commande

Au centre des préoccupations de l'automatique, le problème de la synthèse de la commande vis-à-vis d'incertitudes structurées a suscité un intérêt considérable durant ces dernières décennies. Ce problème est toujours ouvert, puisqu'il n'existe pas actuellement d'approche permettant de garantir des performances du système et la facilité des calculs. Généralement, et dans presque tous les cas, la prise en considération de l'incertitude impose nécessairement un degré de complexité assez important [Bernussou & Oustaloup, 2001].

1.2.4 La robustesse et les incertitude

Le problème de robustesse dans le cas des incertitudes paramétriques a suscité un grand intérêt de la part des automaticiens. Un système de commande est dit robuste s'il garantit des performances correctes malgré l'incertitude de modélisation et les perturbations affectant le processus à commander. La robustesse est mesurée vis-à-vis des incertitudes de nature et d'origine différentes.
D'après [Daafouz, 1997 ; Garcia, 1999], l'origine des incertitudes dans un système est liée à la connaissance imparfaite des paramètres du modèle obtenu, les approximations faites lors de la modélisation du procédé ainsi que la présence de certains phénomènes physiques qui ne sont pas pris en compte par le modèle. Les incertitudes peuvent aussi refléter des perturbations externes intervenant sur les entrées et les sorties du processus et influencer son comportement. On peut noter dans ce cas les bruits de mesure et des perturbations de charge.

1.3 Commande robuste

1.3.1 Principe et concepts de base

La théorie de la commande robuste a attiré l'attention des chercheurs comme une thématique en évolution durant ces dernières années. Elle constitue un outil solide pour la conception des systèmes industriels ainsi que divers problèmes pratiques de l'automatique. [Bernussou, 1996 ; Rotella & Fourquet, 2001 ; Chouiter, 1997 ; Bonnet, 2008 ; Zhou & Doyle 1998].

L'importance de la commande robuste revient à réduire l'influence des effets extérieurs

sur la réponse du système à commander. Généralement, les systèmes dynamiques sont sensibles aux perturbations de charge, aux incertitudes de modélisation, aux saturations d'actionneurs et aux bruits de mesure. La commande robuste cherche à atteindre plusieurs objectifs, tels que la stabilité et la performance des modèles nominaux et incertains.

1.3.2 Techniques de commande robuste

1.3.2.1 La commande $\mathcal{H}_\infty$

La théorie de la commande $\mathcal{H}_\infty$ a été conçue par G. Zames [Zames, 1981], elle n'a connu un réel essor qu'à la fin des années 80, grâce aux travaux de J.C. Doyle [Doyle et al., 1989]. La figure 1.2 montre une représentation sous forme standard de la synthèse

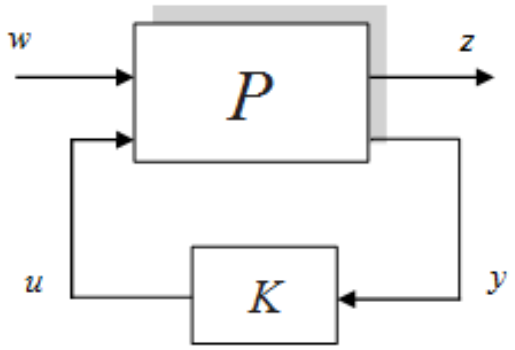

FIGURE 1.2 – Forme standard de la synthèse $\mathcal{H}_\infty$.

d'un système dynamique, dans lequel on note u les entrées de commande, y les sorties mesurées, z les sorties à contrôler et w les entrées exogènes. Les blocs P et K présentent respectivement le modèle nominal pondéré et le correcteur $\mathcal{H}_\infty$ recherché. Après le calcul de la fonction de transfert en boucle fermée entre l'entrée w et la sortie z, la transformation dite fractionnaire linéaire $F_l(P, K)$ est donnée par l'équation suivante :

$$T_{wz}(s) = P_{zw}(s) + P_{zu}(s)K(s)(I_n - P_{yu}K(s))^{-1}P_{yw}(s) \tag{1.1}$$

La synthèse $\mathcal{H}_\infty$ consiste essentiellement à chercher une loi de commande qui assure la stabilité interne du système et détermine la plus petite valeur de la norme $\mathcal{H}_\infty$ du système bouclé, noté $\|F_l(P, K)\|_\infty$ [Kwakernaak, 1993 ; Duc & Font., 1999 ; Zhou & Doyle, 1998 ; Haggège et al., 2009].

1.3.2.2 La commande mixte $\mathcal{H}_2/\mathcal{H}_\infty$

Pour cette approche de commande et comme cela est montré dans la figure 1.3, le système possède deux entrées exogènes w_2 etw_∞ et une entrée de commande u ainsi que, en sorties, z_2, z_∞ et y constituant la mesure. La sortie et les entrées w_2 et z_2 sont utilisées pour définir la norme $\mathcal{H}_2$ du système, tandis que les variables z_∞ et w_∞ sont relatives à la norme $\mathcal{H}_\infty$ [Arzelier et al., 2002 ; Ho et al., 2005]. La commande mixte $\mathcal{H}_2/\mathcal{H}_\infty$ permet de

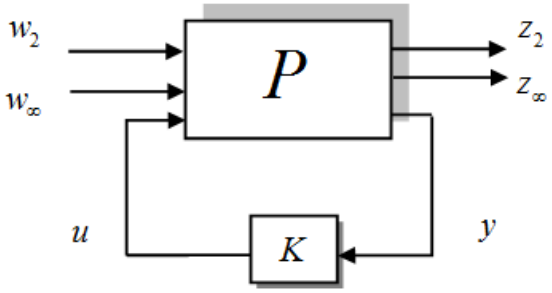

FIGURE 1.3 – Forme standard de la synthèse mixte $\mathcal{H}_2/\mathcal{H}_\infty$.

garantir les performances nominales de la boucle fermée caractérisées par la norme $\mathcal{H}_2$, sous contrainte de stabilité robuste exprimée par la norme w_∞. Le problème de synthèse de la commande mixte $\mathcal{H}_2/\mathcal{H}_\infty$ reste toujours ouvert et seuls quelques résultats analytiques sont disponibles à ce propos [Rotea & Khargonekar, 1991].

1.3.2.3 La commande adaptative

La commande adaptative a été introduite en 1950 pour pallier au problème de l'insuffisance des commandes classiques. Le besoin d'une commande qui permet de répondre aux exigences demandées conduit à proposer cette approche qui traduit le comportement du système. Dans plusieurs applications, les paramètres des systèmes varient au cours du temps, ce qui ne garantit pas la robustesse des régulateurs à paramètres fixes [Astrom & Wittenmark, 1995 ; Borne et al., 1993b ; Salima, 2009 ; Landau & Dugard, 1996 ; Feng & Lozano, 1999 ; Landau et al., 1998]. Ainsi, tout changement au niveau des paramètres du système influe directement sur la performance des régulateurs. Face à ce problème, la commande adaptative incorpore un ensemble de techniques qui consistent à estimer les paramètres des régulateurs pour assurer la performance du système à contrôler.

La commande adaptative est structurée selon trois catégories : la commande adaptative

avec modèle de référence, l'approximation des stratégies de commande optimale stochastique et les systèmes de commande auto-ajustables. La première structure de commande adaptative avec modèle de référence a été proposée par Whittaker, Yamron et Kezer [Whitaker et al., 1958]. Elle est basée sur une stratégie d'identification [Landau et al., 1998]. La deuxième structure a été proposée par Feldbaum en 1963. Cette approche est caractérisée par la complexité et la puissance de calcul requise. Tous les paramètres inconnus sont considérés comme des états additionnels du système qui transforment le problème de commande linéaire en un autre non linéaire. Enfin, la structure de commande adaptative auto-ajustable a été proposée en 1958 par Kalman. Cette approche est basée sur un mécanisme d'adaptation qui permet d'avoir les paramètres auto-ajustés à chaque instant.

En 1970, Aström et Wittenmark ont proposé une nouvelle structure de régulateurs auto-ajustés [Astrom & Wittenmark, 1995]. Deux versions de cette technique de commande sont essentiellement proposées. On cite la commande adaptative directe, dans laquelle les paramètres du régulateur sont directement adaptés, moyennant un algorithme d'adaptation paramétrique approprié, Figure 1.4, et la commande adaptative indirecte, Figure 1.5. La phase de détermination des paramètres du régulateur à partir de ceux du modèle de commande est ainsi contournée. Tous les objectifs de commande, pour lesquels il est possible de décrire le comportement entrée-sortie du système sous une forme linéaire par rapport aux paramètres du régulateur, peuvent être considérés. Le traitement des données et la supervision sont utilisés pour les mêmes considérations que la commande adaptative indirecte, deuxième version de la structure d'Astrom et Wittermark proposée.

Pour la commande adaptative indirecte, Figure 1.5, un modèle de comportement entrée-sortie du système à commander est continûment mis à jour et est utilisé pour la synthèse du régulateur, comme s'il était le meilleur modèle de commande que l'on aurait utilisé. Les paramètres du régulateur sont ainsi adaptés de manière à réaliser les performances requises et une estimation des paramètres du processus par une procédure d'identification est nécessaire. Le bloc de supervision permet d'assurer l'intégrité du système de commande, conformément aux résultats de stabilité, de convergence et de robustesse disponibles dans la littérature [Salima, 2009 ; Landau & Dugard, 1996].

1.3.2.4 La commande prédictive

Proposée à l'origine par J. Richalet [Richalet, 1993 ; Richalet et al., 1978 ; Richalet et al., 1976], la commande prédictive à base de modèle MPC n'a connu un réel essor qu'à la fin des années 80, grâce aux travaux de D.W. Clarke [Clarke et al., 1987]. Ces travaux ont

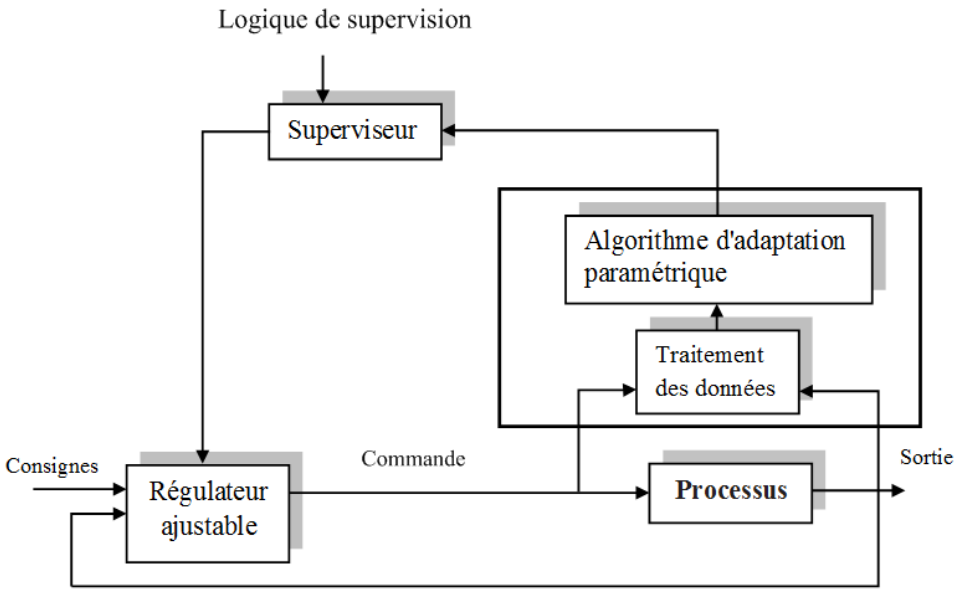

FIGURE 1.4 – Principe de la commande adaptative directe.

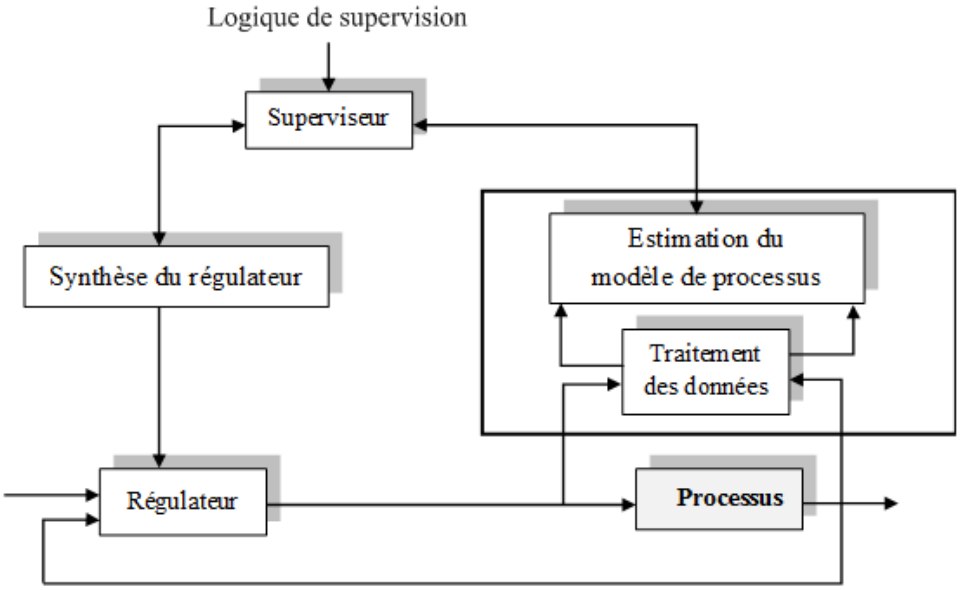

FIGURE 1.5 – Principe de la commande adaptative indirecte.

généralisé cette technique de commande qui est ensuite devenue la commande prédictive généralisée GPC. Le principe de la commande GPC est basé sur l'utilisation d'un modèle numérique du processus à commander pour la prédiction des sorties futures sur un horizon déterminé, et ensuite sur la synthèse d'un système de commande capable d'anticiper les variations de la consigne, figure 1.6.

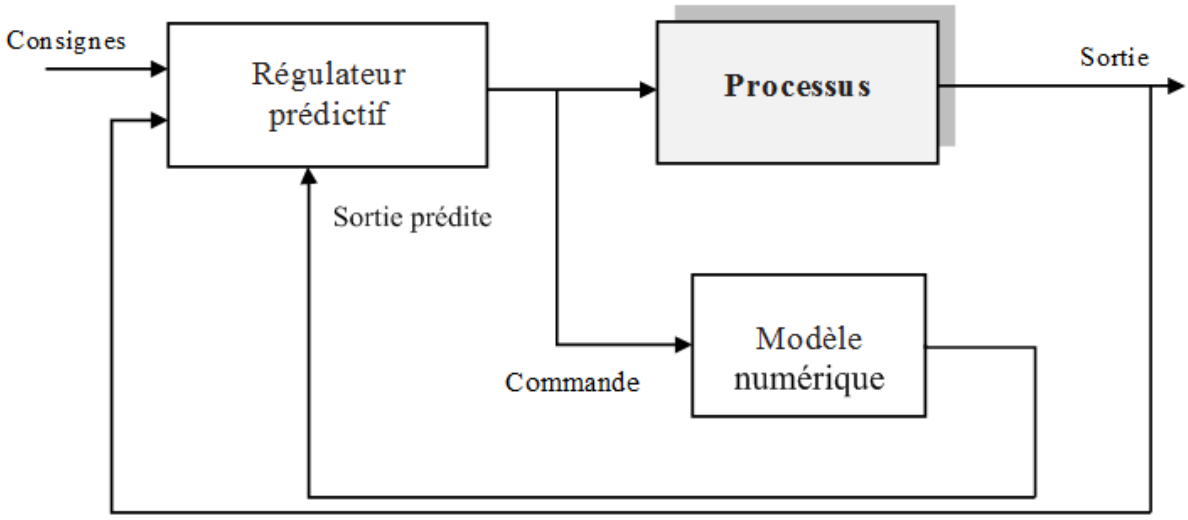

FIGURE 1.6 – Principe de la commande prédictive GPC.

1.3.2.5 La commande multi-modèle

L'approche multi-modèle est une méthode de commande basée sur la modélisation multi-modèle des systèmes complexes et non linéaires et présentant un bon compromis entre la complexité du modèle et les performances attendues. Cette approche [Murray-Smith & Johansen,1997 ; Delmotte, 1997] constitue même une alternative intéressante, très utilisée actuellement pour la modélisation des systèmes non linéaires.

Le principe du multi-modèle est basé sur la décomposition du comportement global du processus dynamique en un nombre fini de zones de fonctionnement partiel. Chaque zone est caractérisée par un sous-modèle du processus, dit modèle local. L'ensemble de ces sous-modèles, représentant une formulation mathématique simplifiée du système, forme ainsi une bibliothèque de modèles qui sert à élaborer la loi de commande. Un mécanisme de décision spécifié, par commutation ou par fusion, est adopté afin de tester la validation de chacun des modèles de la bibliothèque. Cette formulation donne alors naissance aux régulateurs multi-modèles ou plus généralement à la multi-commande, dont le principe est

illustré dans la figure 1.7. Dans la littérature, des terminologies équivalentes sont utilisées pour définir ce type de modèles. On parle ainsi de modèles flous de Takagi-Sugeno [Takagi & Sugeno, 1985],de modèles linéaires polytopiques (PLM) [Angelis, 2001], etc.

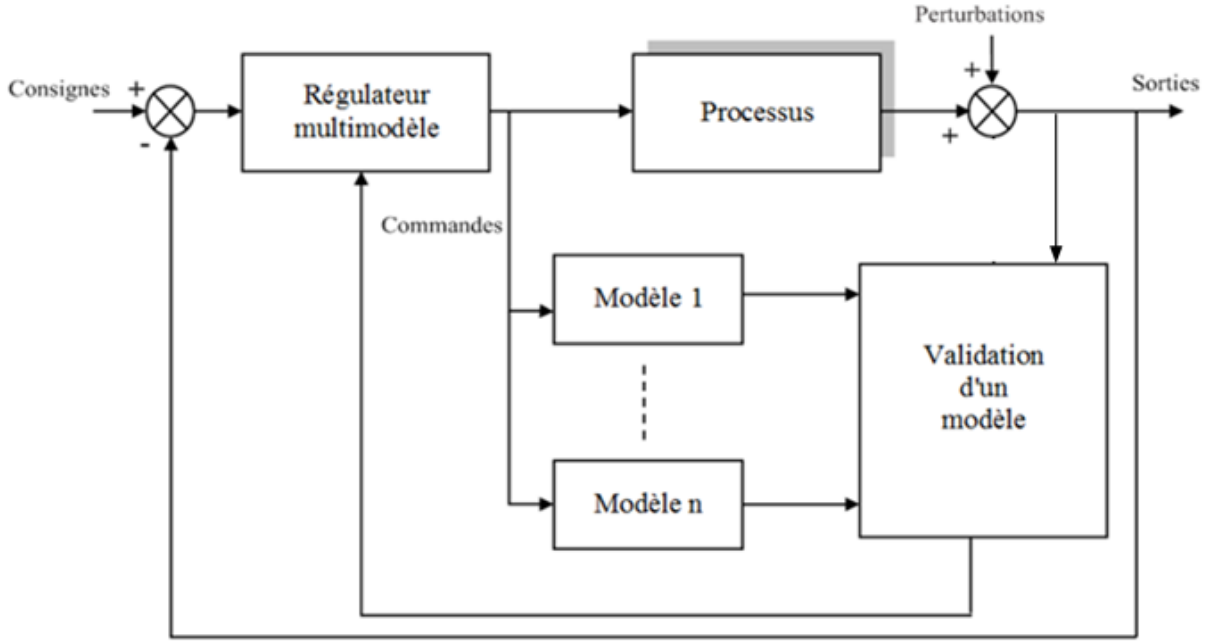

FIGURE 1.7 – Principe de la commande multi-modèle.

1.3.2.6 La commande CRONE

Proposée en 1975 par A. Oustaloup [Oustaloup, 1975], la commande CRONE a connu un véritable essor au cours de ces dernières décennies. Cette technique de commande utilise les développements avancés en théorie de la modélisation des systèmes fractionnaires utilisant les concepts de la dérivation non entière. La mise en oeuvre pratique de cette théorie de commande a connu un succès remarquable dans de vastes domaines industriels, et plus particulièrement dans l'industrie automobile, grâce au principe de la suspension CRONE.

Le principe est d'assurer la robustesse du degré de stabilité de la commande vis-à-vis des incertitudes du procédé. Le degré de stabilité est mesuré par le facteur de résonance en asservissement ou le facteur d'amortissement en asservissement et en régulation. L'élaboration de la loi de commande CRONE est basée sur la manipulation des marges de stabilité et plus particulièrement la marge de phase.

Selon la génération de la commande, la synthèse du régulateur est effectuée pour la trans-

mittance de la boucle ouverte d'ordre non entier [Oustaloup, 1995 ; Oustaloup et al., 1999].

1.3.2.7 La commande polynômiale à structure RST

Cette structure de commande est fondée sur la synthèse d'un régulateur numérique à deux degrés de liberté dit RST [Landau et al., 1998 ; Landau, 1998 ; Landau & Karimi, 1998] conformément à la figure 1.8. Les signaux $r(t)$, $u(t)$, $y(t)$, $p_1(t)$ et $p_2(t)$ désignent respectivement les grandeurs de consigne, de commande, de sortie, de perturbation et de bruit de mesure.

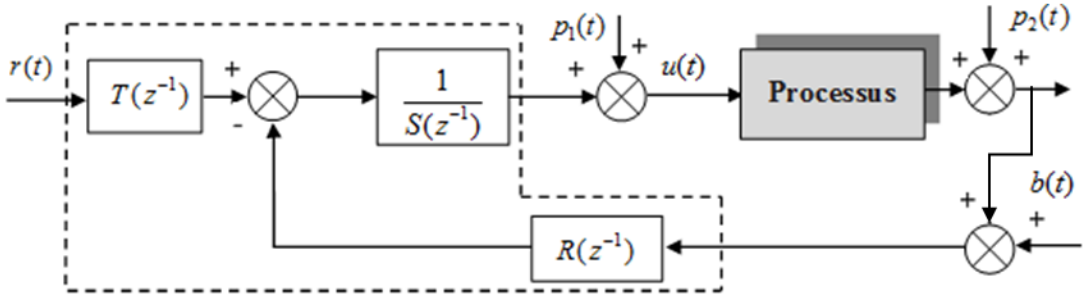

FIGURE 1.8 – Structure de la commande polynômiale RST.

Le régulateur RST est un organe de contrôle permettant d'imposer séparément un comportement de la poursuite et de régulation en boucle fermée d'un système. Ce contrôleur est couramment utilisé dans les systèmes de commande numérique. Le sigle RST vient du nom des trois polynômes R, S et T , qui doivent être déterminés afin d'obtenir la loi de commande numérique récurrente [Landau et al., 1998]. La synthèse et la conception des régulateurs robustes de type RST sont basées principalement sur les méthodes de placement de pôles et de calibrage des fonctions de sensibilité [Prochazka & Landau, 2002]. La robustesse des régulateurs RST vis-à-vis de gabarits fréquentiels traduit l'incertitude des paramètres, les perturbations et les bruits de mesure. Pour cela, les caractéristiques fréquentielles des modules de fonctions de sensibilité sont calibrées afin de satisfaire de tels gabarits [Prochazka & Landau, 2003 ; Varga , 2000 ; Langer & Landau, 1999].

1.3.2.8 La commande par logique floue

Les théories à la base de la logique floue ont été établies par L. Zadeh en 1965 [Zadeh, 1965]. La puissance de la logique floue se voit à partir de sa capacité à décrire

des phénomènes et des processus de façon linguistique et à les concevoir en plusieurs règles. La commande par logique floue a été appliquée à l'origine par E. H. Mamdani en 1974 [Mamdani, 1974]. Elle permet d'avoir une loi de réglage efficace. Contrairement aux régulateurs standards ou à contre réaction, le régulateur par logique floue, en abrégé RLF, ne traite pas des relations mathématiques, par contre il n'utilise que des inférences avec des règles basées sur des variables linguistiques. La figure 1.9 présente le schéma synoptique de base d'un tel régulateur. Un RLF est basé sur quatre blocs principaux : la base de connaissance, la fuzzification, le moteur d'inférence et la défuzzification [Lee, 1990a, 1990b ; Passino & Yurkovich, 1998]. Le rôle global du RLF est de convertir les

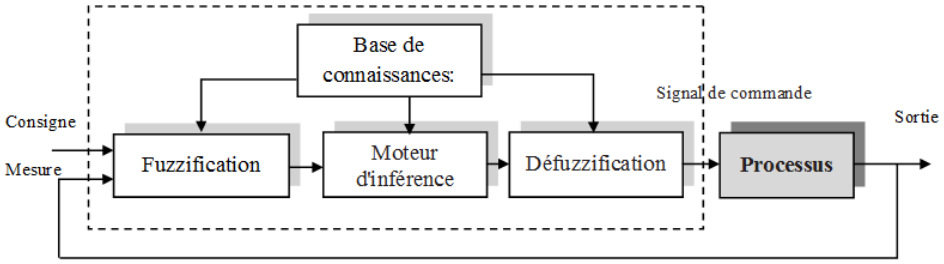

FIGURE 1.9 – Synoptique d'un régulateur par logique floue.

valeurs d'entrée dits déterministes en des valeurs floues, puis de les traiter moyennant des règles floues. Enfin, on procède à la conversion du signal de commande sous forme de valeurs floues en valeurs déterministes pour pouvoir l'appliquer au processus. Le bloc de fuzzification permet de convertir les données d'entrée en valeurs linguistiques pouvant être manipulées par le mécanisme d'inférence. Le bloc de défuzzification effectue une transformation qui fournit un signal de commande numérique à partir du signal flou déduit. L'inférence est le bloc qui permet de déduire les actions de la commande floue à l'aide des règles d'inférence dans la logique floue. Le bloc de connaissances est constitué d'une base de données et d'une base de règles dont la première a pour rôle d'établir les règles de commande et de manipuler les règles dans le régulateur et la deuxième présente une stratégie de commande.

1.3.2.9 La commande neuronale

Le premier modèle de neurone formel a été proposé par W. McCulloch et W. Pitts en 1943. Inspiré du neurone biologique, le neurone formel présente une fonction algébrique dont les valeurs dépendent de entrées appelées poids synaptiques, ou aussi poids de

connexion [Borne et al., 2007].

Un réseau de neurones est un ensemble d'éléments de calcul (neurones) interconnectés avec une loi d'apprentissage qui rend possible d'ajuster les paramètres du réseau, Figure 1.10. Vu leur capacité d'approximation, d'adaptation et d'apprentissage, l'utilisation des réseaux de neurones a connu un réel essor pour la conception de stratégies de commande des systèmes dynamiques [Haggège, 2003 ; Borne et al., 2007 ; Nerrand et al., 1992]. La synthèse d'un système de commande neuronale porte essentiellement sur la construction d'un réseau de neurones et sur la réalisation de son apprentissage pour le pilotage du procédé à commander. Différentes stratégies de commande neuronale sont été proposées dans la littérature. Nous pouvons citer la commande neuronale directe par modèle inverse, la commande neuronale par modèle interne, la commande basée sur l'erreur de sortie et la commande adaptative. Les stratégies de commande utilisant les réseaux de neurones

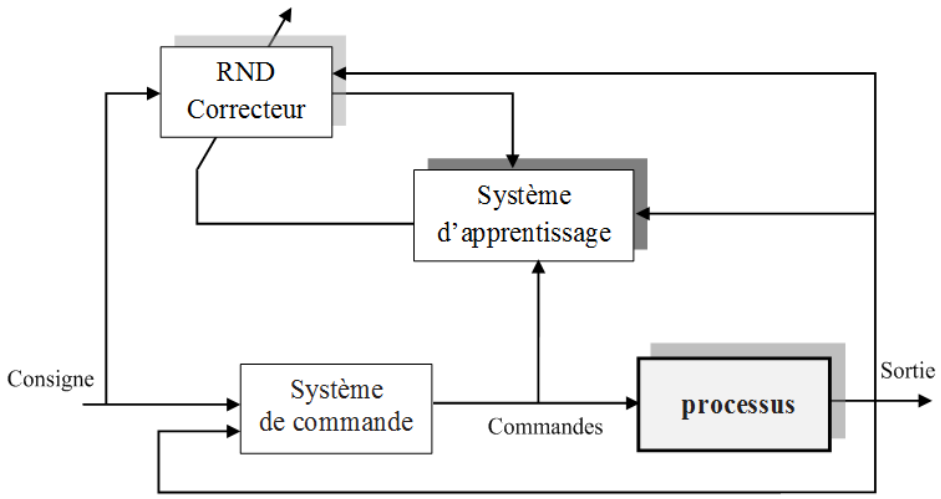

FIGURE 1.10 – Exemple d'une structure de commande neuronale en phase d'apprentissage.

ont certaines limitations, telles que l'absence d'une technique systématique permettant de donner a priori le nombre de neurones dans la couche cachée. On peut citer aussi le problème des valeurs initiales des poids du réseau ; en effet, plus la somme pondérée des entrées d'un neurone est élevée, plus le neurone se trouve dans la zone de saturation de sa fonction d'activation, donc plus sa dérivée est faible. D'autres problèmes restent largement ouverts, comme les problèmes de sur-apprentissage et de réglage du pas d'apprentissage [Borne et al., 2007].

1.3.2.10 La commande neuro-floue

La jointure entre les réseaux de neurones et la logique floue a permis de mixer et de tirer les avantages des deux approches. Nous parlons ainsi de techniques neuro-floues. Les approches neuronales et floues présentent respectivement la capacité d'apprentissage et de lisibilité [Haggège, 2003]. Cette fusion a été proposée depuis 1988, en tenant compte des limitations des deux approches séparément. Malgré cette capacité d'apprentissage, les réseaux de neurones ont toujours une limitation remarquable au niveau de leurs paramètres de contrôle et leurs structures ; en outre, les connaissances humaines ne peuvent être jamais utiles pour les construire. Par contre, le moteur d'inférence floue présente deux points forts par rapport aux réseaux de neurones, situés au niveau de l'exploitation des connaissances humaines et la capacité puissante de description due à l'utilisation des variables linguistiques [Borne et al., 2007 ; Haggège et al., 2001a, 2001b ; Haggège & Benrejeb, 2000]. Deux approches neuro-floues existent [Baghli et al., 1997], La première méthode consiste à concevoir le moteur d'inférence floue sous la forme d'un réseau de neurones. Cette architecture est sensible aux types des règles, aux méthodes d'inférence, à la défuzzication et à la fuzzification choisis. La deuxième approche consiste à utiliser les réseaux de neurones pour remplacer chacune des composantes d'un système de commande floue. La figure 1.11 présente une structure de commande neuro-floue à apprentissage en ligne [Haggège & Benrejeb, 2000 ; Haggège et al., 2001a].

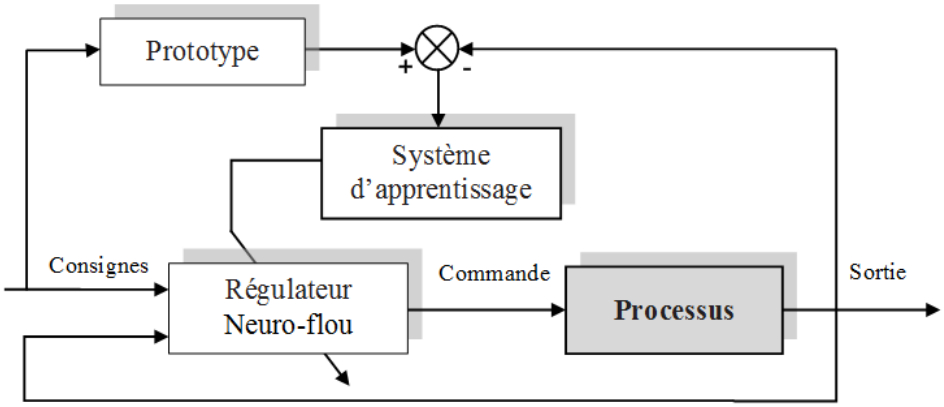

FIGURE 1.11 – Exemple d'une structure de commande neuro-floue.

1.4 Complexité de la synthèse et recours à l'optimisation

Dans cette section, nous nous intéresserons aux problèmes liés à la synthèse de lois de commande robuste. Quelques exemples issus de la littérature seront cités et discutés.

1.4.1 Problèmes complexes de synthèse de commande robuste

1.4.1.1 Synthèse de régulateurs RST

Appelée aussi problème de placement de pôles en boucle fermée, la synthèse des régulateurs RST est basée sur le choix délicat des pôles à placer en boucle fermée afin d'assurer les objectifs de poursuite et de régulation. D'après Landau et al. [Landau & Karimi, 1998 ; Landau et al., 1998], le calibrage des fonctions de sensibilité présente une approche de synthèse des régulateurs robustes RST basée sur une procédure itérative d'essais-erreurs. Ces fonctions de sensibilité permettent d'évaluer la stabilité et la robustesse du système en boucle fermée et de quantifier les incertitudes tolérées dans les différentes régions fréquentielles.

1.4.1.2 Synthèse et réglage des contrôleurs flous

Le réglage de la commande floue est basé sur une procédure d'essais-erreurs coûteuse en temps de conception et souvent délicate pour l'obtention des réponses désirées et souhaitables du système. La conception et l'implémentation de ce type de commande deviennent très compliquées avec la complexité du procédé à contrôler.

1.4.1.3 Synthèse de la commande $\mathcal{H}_\infty$

Les méthodes classiques de synthèse , utilisant des équations algébriques de riccati [Doyle et al., 1989] ou des inégalités matricielles linéaires [Gahinet & Apkarian, 1994 ; Iwasaki & Skelton, 1994], conduisent à des correcteurs d'ordre complet. Cet ordre est égal à celui du système augmenté de filtres de pondération, censés traduire les spécifications de performances et de robustesse du cahier des charges. La réduction de l'ordre des correcteurs est souvent adoptée comme procédure classique [Zhou & Doyle 1998 ; Zhou et al., 1996]. Cette solution conduit le plus souvent à une augmentation de la norme du système bouclé, et par conséquent des dégradations en performances et en robustesse de la commande élaborée.

1.4.2 Solution proposée

Devant ces difficultés rencontrées, plus précisément dans la phase de synthèse de la loi de commande robuste, les automaticiens trouvent que la théorie de l'optimisation peut présenter une solution efficace et prometteuse pour ces problèmes NP difficiles. Dans la littérature, il existe plusieurs techniques d'optimisation robustes et performantes en termes de temps de calcul et de qualité des solutions fournies. Dans cette thèse, on s'intéresse aux méthodes approchées dites "métaheuristiques".

1.4.3 Théorie de l'optimisation continue dynamique

1.4.3.1 Principe général

La recherche d'un état de fonctionnement et de production dit optimal est parmi les soucis majeurs des ingénieurs dans divers domaines. Nombreux sont les problèmes scientifiques qui peuvent être traités de façon qu'on puisse ajuster et améliorer leurs paramètres pour produire des résultats fiables. L'orientation vers l'optimisation est une solution pour résoudre plusieurs problèmes et sujets, en particulier ceux relatifs à la théorie de la commande. Dans la littérature, l'optimisation est définie comme un outil pour la sélection des meilleures solutions nommées optimales. Ce domaine est inspiré des mathématiques et il est aujourd'hui utilisé en informatique et dans d'autres domaines comme l'automatique et la mécanique. Parmi les premières méthodes reposant sur des résolutions mathématiques, on peut citer les travaux de Lagrange, de Hamilton et de Fermat. Aussi, d'autres techniques qui utilisent des algorithmes itératifs ont été proposées par Newton et Gauss. Historiquement, ces méthodes ont été fréquemment utilisées pendant la deuxième guerre mondiale pour des programmes militaires. Généralement, les problèmes d'optimisation sont connus comme une recherche d'un minimum ou d'un maximum, qui présente la solution cherchée ou l'optimum d'une fonction donnée. Dans certains cas, les variables d'une telle fonction à optimiser sont contraintes à évoluer dans l'espace de recherche. Il s'agit ainsi d'un problème d'optimisation sous contraintes [Coello Coello et al., 2007 ; Dixon & Szego 1975 ; Dixon & Szego, 1978]. Pour que le fonctionnement d'un système obéisse au cahier des charges présenté, on devra le calibrer de manière à :

- occuper le volume minimum nécessaire pour son bon fonctionnement (coûts des matières premières) ;
- consommer le minimum d'énergie (coûts de fonctionnement) ;
- répondre à la demande de l'utilisateur (cahier des charges).

Pour cela, on distingue plusieurs critères pour classifier les problèmes d'optimisation, tels que le nombre de variables de décision, le type de variables de décision, les propriétés mathématiques de la fonction objectif et la formulation du problème à optimiser. Ainsi, un problème d'optimisation sous contraintes est défini sous la forme générique suivante [Coello Coello et al., 2007] :

$$\underset{\substack{x\in D\subseteq\Re^m \\ \text{sous contraintes}}}{\text{minimiser}} \quad f_i\,(x)\,, \qquad \underset{\substack{h_j(x)=0,\ (j=1,2,...,J),\\ g_k(x)\leq 0,\ (k=1,2,...,K),}}{(i=1,\,2,\,\ldots,M)}\,, \tag{1.2}$$

où $x = (x_1, x_2, ..., x_m)$ représentent les variables de décision, $f_i : \Re^m \to \Re$ la i-ème fonction objectif à minimiser, $g_k : \Re^m \to \Re$ et $h_j : \Re^m \to \Re$ les contraintes du problème et $D = \{x \in \Re^m; x_{\min} \leq x \leq x_{\max}\}$ l'espace de recherche (ou espace de décision) supposé borné. Les problèmes d'optimisation ont pour but de chercher les meilleures solutions, c'est-à-dire l'optimisation globale. Cependant, il peut exister d'autres solutions intermédiaires, qui sont également des solutions optimums, mais uniquement pour un sous-espace restreint, c'est l'optimisation locale.

Définition 1.1 : Optimisation locale

Un minimum local de la région B , noté x^*_B est défini comme :

$$f(x^*_B) \leq f(x), \ \ \forall x \in B \tag{1.3}$$

Un espace de recherche D donné peut contenir plusieurs régions B_i telles que $B_i \cap B_j$ si $i \neq j$. Il s'ensuit donc que $x_i = x_j$, de sorte que le minimum de chaque région est unique B_i.Tous les x_i peuvent être considérés comme des minimums de B , même s'ils ne sont que des limiteurs locaux. La valeur $f(x^*_{B_i})$ sera appelée le minimum local.

La plupart des algorithmes d'optimisation nécessitent un point de départ $z_0 \in D$. Un algorithme d'optimisation locale devrait garantir qu'il sera en mesure de trouver le minimum $x^*_{B_i}$ de l'ensemble B si $z_0 \in B$. De nombreux algorithmes d'optimisation locale ont été proposés. Les algorithmes déterministes d'optimisation locale s'articulent autours des algorithmes de Newton-Raphson, de Fletcher-Reeves(FR), de Polar-Ribiere (PR), de Davidon-Fletcher-Powell(DFP) et de Broyden-Fletcher-Goldfarb-Shanno (BFGS) [Polak, 1997 ; Bishop, 1995].

Définition 1.2 : Optimisation globale

Un minimum global x^*_B de la région B est défini comme :

$$f(x^*) \leq f(x), \ \ \forall x \in \ D \tag{1.4}$$

Pour des problèmes non contraints, il est courant de choisir $D = \Re^m$, m étant la dimension de x. Tout au long de cette étude, le terme d'optimisation globale se réfère strictement au processus de recherche de x^* comme défini dans (1.4). Le terme minimum global se référera à la valeur $f(x^*)$. Un algorithme d'optimisation globale, comme les algorithmes d'optimisation locale décrits ci-dessus, procède également en choisissant la position initiale $z_0 \in D$. Ils sont en mesure de trouver un minimum local de $B \subset D$, quelle que soit la position réelle de z_0 . Ces algorithmes sont constitués de deux processus : étape "globale" et étape "locale". Leurs étapes locales sont habituellement l'application d'un algorithme local de minimisation et leurs étapes "globales" sont conçues pour s'assurer que l'algorithme entrera dans une région B_i, d'où l'étape "locale" pourra trouver le minimum du B_i.Ces méthodes seront appelées algorithmes globalement convergents, ce qui signifie qu'ils sont capables de converger vers un minimum local, peu importe leur position de départ z_0. Ces méthodes sont également capables de trouver le minimum global, étant donné que la position de départ z_0 est bien choisie [Dixon & Szego, 1978 ; Dixon & Szego, 1975]. La figure 1.12 illustre la différence entre le minimum local x_B^* et le minimum global x^* .

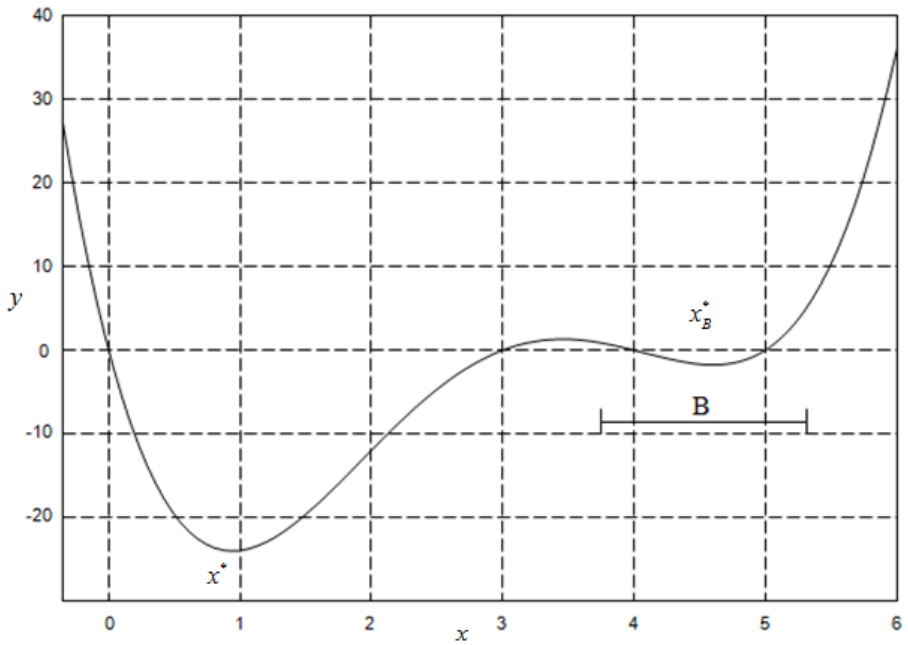

FIGURE 1.12 – La fonction $f(x) = x^4 - 12x^3 + 47x^2 - 60x$, indiquant le minimum global x^* ainsi que le minimum local x_B^*.

1.4.3.2 Classification des méthodes d'optimisation

Comme montré dans [Collette & Siarry, 2004], il existe de nombreuses méthodes d'optimisation que l'on peut classifier de la façon indiquée dans la figure 1.13.

On différencie en premier lieu l'optimisation continue et l'optimisation combinatoire appelée aussi discrète, figure 1.13. Pour les problèmes d'optimisation combinatoire (à variables discrètes), les méthodes dites exactes sont des techniques qui permettent d'obtenir des solutions optimales dans un temps de calcul acceptable. Contrairement aux heuristiques, que nous présentons dans la section suivante, toutes les méthodes exactes obtiennent une solution optimale et non une solution approchée. Il existe plusieurs méthodes exactes, dites déterministes, qui permettent de résoudre certains types de problèmes d'optimisation, sachant que les fonctions objectifs sont convexes, continues ou encore dérivables. Basés sur ce principe, on peut citer plusieurs algorithmes comme la méthode du simplexe de Dantzig [Dantzig, 1963 ; Nelder & Mead, 1965], la méthode de back-tracking aussi nommée méthode de retour sur trace, la méthode de programmation linéaire [Schrijver, 1998], quadratique [Nocedal & Wright, 2000] ou dynamique [Bertsekas, 1995] ou encore la méthode du gradient [Avriel, 1976].

Les méthodes approchées sont conçues pour des problèmes plus complexes ou de taille importante. La confrontation de ce genre de problèmes est possible avec une heuristique ou une métaheuristique. Parmi les métaheuristiques, on peut différencier les métaheuristiques de "voisinage" qui font progresser une seule solution à la fois et celles "distribuées" qui manipulent en parallèle toute une population de solutions. Les méthodes basées sur la recherche locale consistent à faire évoluer une seule solution, en améliorant progressivement sa qualité dans un voisinage spécifique. Pour ces méthodes, la recherche s'arrête lorsqu'une solution localement optimale est trouvée. Ces méthodes sont appelées aussi méthodes d'amélioration itérative, partant d'une solution initiale, et ont pour but de l'améliorer au cours des itérations.

La meilleure solution est toujours trouvée dans le voisinage. On arrête les itérations si on ne parvient pas à améliorer la solution finale. Différentes méthodes de recherche locale existent dans la littérature est peuvent être déterministes comme la méthode du gradient et d'autres, stochastiques, comme la recherche Tabou, la méthode GRASP, la recherche locale guidée.

Contrairement aux méthodes partant d'une solution singulière, les méthodes à popu-

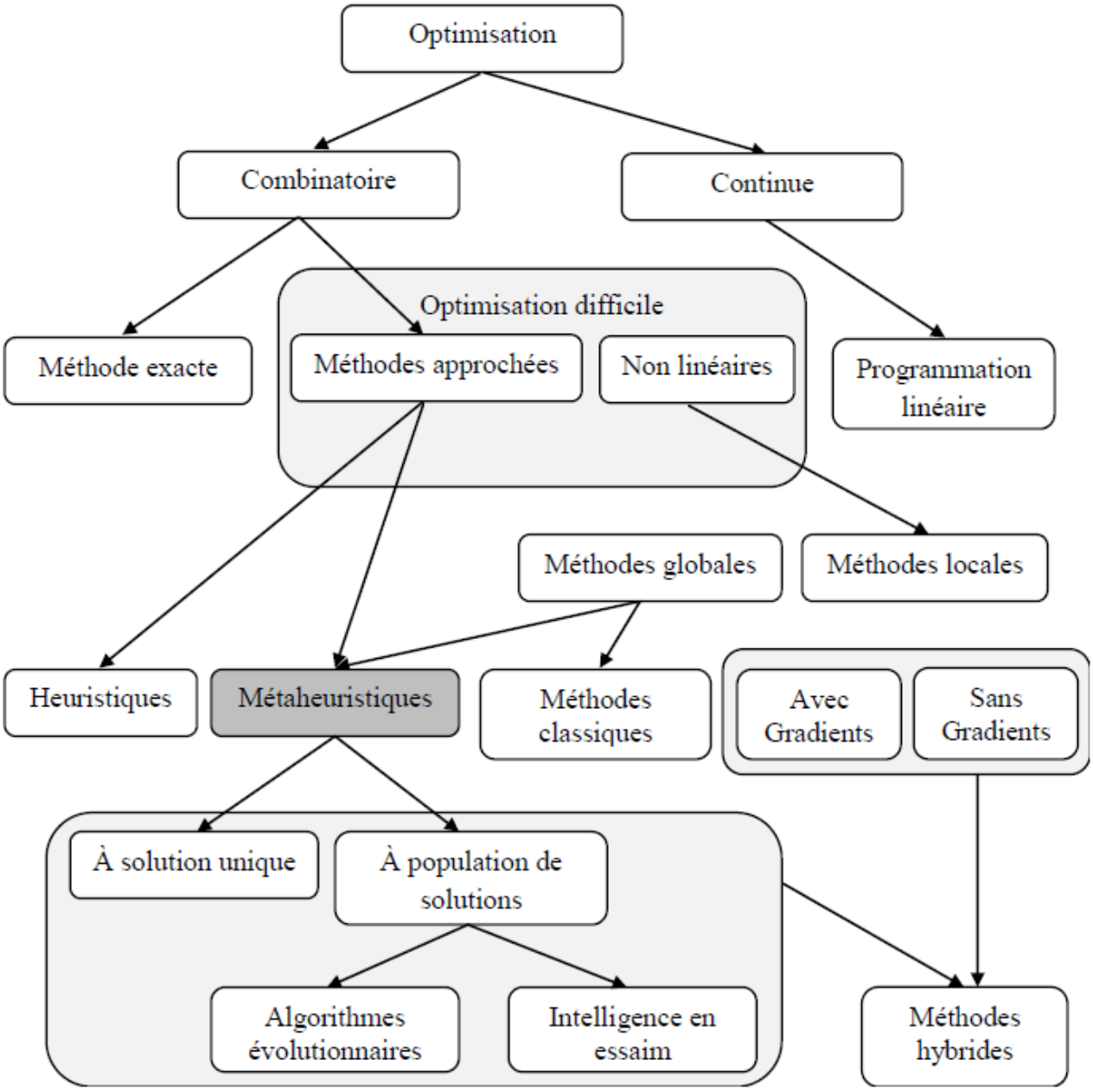

FIGURE 1.13 – Classification possible des différents types de méthodes d'optimisation [Collette & Siarry, 2004].

lation de solutions font évoluer un ensemble de solutions initiales, plutôt qu'une seule solution à chaque itération. Parmi ces méthodes, on peut citer les algorithmes génétiques, les essaims particulaires et les algorithmes de colonies de fourmis. Il est possible d'intégrer dans une méthode d'optimisation deux méthodes distinctes (métaheuristiques, heuristiques ou méthodes exactes), on parle alors de méthodes hybrides. Le but de cette com-

binaison est de tirer les avantages des différentes approches.

1.4.3.3 Heuristiques et métaheuristiques

Comme on a mentionné dans la section précédente, les méthodes d'optimisation peuvent être stochastiques ou déterministes. Compte tenu de la différence entre les méthodes exactes et les heuristiques et à cause du temps de calcul élevé, l'utilisation des méthodes exactes n'est pas toujours recommandée. Par conséquent, l'utilisation des méthodes approchées, appelées aussi heuristiques, est plus fiable.

Le mot heuristique est d'origine greceurisko qui signifie "je trouve". Une heuristique peut être définie comme une méthode approximative qui fournit une solution acceptable mais pas toujours optimale. La solution approchée est obtenue par ces méthodes en se basant sur la structure du problème d'optimisation considéré. Cette solution est de qualité raisonnable en un temps réduit de recherche.

Le terme métaheuristique a été inventé par Fred Glover lors de la conception de la méthode recherche Tabou [Feo & Resende, 1989]. Les métaheuristiques sont des méthodes qui permettent de chercher une solution optimale en explorant l'espace de recherche efficacement pour trouver une solution presque optimale.

1.5 Métaheuristiques pour l'optimisation difficile

1.5.1 Généralités et concepts de base

Les métaheuristiques sont classifiées parmi les techniques stochastiques destinées à la résolution de différents problèmes d'optimisation dans l'ingénierie. Les métaheuristiques ont montré une grande capacité d'adaptation à plusieurs problèmes dans divers domaines. Même au niveau des algorithmes, les ingénieurs ne les pas trop changés, d'où le terme méta qui signifie "à un plus haut niveau". En général les métaheuristiques présentent des algorithmes qui ont un comportement itératif, et qui permettent au schéma de se reproduire un certain nombre de fois au cours de l'optimisation. En outre, les métaheuristiques offrent la possibilité de modifier les paramètres de contrôle pour la majorité des algorithmes, en tenant toujours compte de la complexité des problèmes abordés. Ces algorithmes tirent leur efficacité du fait qu'ils peuvent s'échapper des optima locaux, dont la présence est un problème fréquemment rencontré en optimisation [Borne & Benrejeb, 2010 ; Dréo et al., 2003 ; Clerc & Siarry, 2009 ; Glover, 1986].

On peut distinguer les métaheuristiques à base d'une solution unique qui exploitent l'espace de recherche et d'autres à base de populations de solutions permettant la meilleure diversification de cet espace. Ayant une simplicité remarquable de mise en oeuvre, les métaheuristiques sont efficaces pour des problèmes NP-difficiles. Elles peuvent aussi exploiter l'expérience accumulée durant la recherche de la solution, afin de guider au mieux les processus de recherche. Pour cela, la majorité des algorithmes métaheuristiques contiennent des mécanismes qui permettent d'éviter le blocage dans des régions de l'espace de recherche et d'assurer la fin du processus de recherche. La plupart des métaheuristiques sont inspirées de la nature : de la biologie pour les algorithmes génétiques, de l'éthologie pour les essaims particulaires et les algorithmes de colonie de fourmis et de la physique pour le recuit simulé [Dréo et al., 2003].

Les métaheuristiques sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds de l'algorithme employé. Dépassant le cadre de l'optimisation mono-objectif, les métaheuristiques se prêtent facilement à des extensions, dont les principales portent sur l'optimisation multiobjectif, l'optimisation multimodale, l'optimisation dynamique, l'adaptation aux problèmes à variables continues et aux problèmes bruités, la parallélisation et l'hybridation. Ces métaheuristiques sont désormais d'un emploi très courant dans tous les secteurs de l'ingénierie.

1.5.2 Métaheuristiques à solution unique

Dans cette section, on présente les métaheuristiques à base de solution unique et qui sont appelées aussi les méthodes de trajectoire. Ces algorithmes commencent la recherche avec une seule solution, qui circule en construisant une trajectoire dans l'espace de recherche. Nombreuses sont les méthodes qui s'articulent sur ce principe : on peut citer essentiellement la méthode de recuit simulé, la méthode GRASP, la méthode de descente, la recherche tabou et la recherche locale itérée [Borne & Benrejeb, 2010].

1.5.2.1 Méthode GRASP

La méthode GRASP est une procédure itérative proposée par Thomas A. Feo et Mauricio G.C. Resende en 1989 [Feo & Resende, 1989 ; Feo & Resende, 1995]. Cette méthode est classifiée parmi les méthodes hybrides, car elle tire avantage des méthodes de voisinage, des heuristiques gloutonnes et de la recherche aléatoire. Chaque itération de cet algorithme

est composée de deux étapes : l'étape de construction d'une solution, suivie par une descente pour améliorer la solution construite. A cette étape, et après des itérations, une solution est construite : on ajoute un élément dans la solution partielle courante à chaque itération. Avec une fonction gloutonne, on peut déterminer une liste des meilleurs candidats à ajouter. A cette phase de construction, la liste des meilleurs candidats est dynamiquement mise à jour à chaque itération. La fin de l'étape de construction est marquée par l'obtention d'une solution complète et à partir de cette dernière, une descente est appliquée pour l'améliorer. La procédure GRASP répète l'étape de la construction et celle de la recherche locale puis retourne la meilleure solution obtenue. Cette méthode est appliquée à beaucoup de problèmes d'optimisation, vu qu'elle est basée sur un nombre réduit de paramètres, qui sont la longueur de la liste de candidats et le nombre d'itérations autorisées. Comme nous l'avons mentionné au début, l'étape de construction est la plus importante pour la méthode GRASP. Le pseudo-code pour la construction est le suivant [Feo & Resende, 1995] :

Algorithm 1 GreedyRandomizedConstruction (α)

$Solution \leftarrow \phi$
$C \leftarrow E$
Evaluation du coût incrémental $c(e)$,$e \in C$
Tant que ($C \neq \phi$) **faire**
$C_{\min} \leftarrow \min\{c(e), e \in C\}$
$C_{\max} \leftarrow \max\{c(e), e \in C\}$
$RCL \leftarrow \{e \in C | c(e) \leq C_{\min} + \alpha\,(C_{\max} - C_{\min})\}$
Sélection aléatoire S de depuis RCL
Solution $\leftarrow$Solution $\cup\{S\}$
Mise à jour de C
Réévaluation du coût incrémental pour $e \in C$
Fin Tant-Que
Retourne Solution
FinGreedyRandomizedConstruction

Dans ce pseudo-code, on désigne par la solution possible avec $S \in F$ (ensemble de toutes les solutions possibles) qui est construite à partir des éléments $e \in E$ (ensemble de base) élément par élément suivant l'heuristique semi-gloutonne. Cet algorithme est un algorithme glouton augmenté d'un aspect aléatoire. C est la liste des candidats. Le

coût incrémental $c(e)$ pour chaque élément de C doit être évalué, permettant ainsi de sélectionner les meilleurs éléments et de les stocker dans une autre liste, appelée liste limitée des candidats RCL (Restricted Candidate List), $\alpha \in [0, 1]$ est un paramètre qui contrôle la qualité du RCL, $C_{\min}$, $C_{\max}$ sont respectivement les coûts incrémentaux minimal et maximal depuis la liste C. La phase de la recherche locale est illustrée par le pseudo-code suivant :

Algorithm 2 RechercheLocal(Solution)

1: **Tant que** (Solution non optimal) **faire**
2: Rechercher S' tant que $f(S') \leq f(Solution)$
3: Evaluation du coût incrémental $c(e)$, $e \in C$
4: $Solution \leftarrow S'$
5: **Fin Tant que**
6: Retourne Solution
7: **Fin RechercheLocal**

1.5.2.2 Recuit simulé

La méthode de recuit simulé a pour origine la mécanique statistique [Kirkpatrick et al., 1983 ; Cerny, 1985]. S. Kirkpatrick et ses collègues étaient des spécialistes de physique ; ils s'intéressaient à la configuration de basse énergie de matériaux magnétiques désordonnés, regroupés sous le terme de verres de spin. L'idée était de traiter ce problème en s'inspirant de la technique expérimentale du recuit utilisée par les métallurgistes pour obtenir un état solide d'énergie minimale. Cette technique consiste à porter le matériau à haute température, puis à abaisser lentement sa température [Dréo et al., 2003 ; Siarry & Dreyfus, 1989]. La méthode du recuit simulé transpose le procédé du recuit à la résolution d'un problème d'optimisation. La fonction objectif du problème, analogue à l'énergie d'un matériau, est alors minimisée, moyennant l'introduction d'une température fictive, qui est, dans ce cas, un simple paramètre de contrôle de l'algorithme. En pratique, la technique exploite l'algorithme de Metropolis, qui permet de décrire le comportement d'un système en équilibre thermodynamique à une certaine température T [Metropolis et al., 1953 ; Hastings, 1970 ; Bonomi & Lutton, 1988]. Partant d'une configuration donnée, on fait subir au système une modification élémentaire ; si cette transformation a pour effet de diminuer la fonction objectif (ou énergie) du système, elle est acceptée ; si elle provoque au contraire une augmentation ΔE de la fonction objectif, elle peut être acceptée tout de

même, avec la probabilité $e^{\left(\frac{-\Delta E}{T}\right)}$. On itère ensuite ce procédé, en gardant la température constante, jusqu'à ce que l'équilibre thermodynamique soit atteint, concrètement au bout d'un nombre "suffisant" de modifications. On abaisse alors la température, avant d'effectuer une nouvelle série de transformations. La loi de décroissance par paliers de la température est souvent empirique, tout comme le critère d'arrêt du programme. Pour cette métaheuristique, les deux mécanismes de diversification et d'intensification sont contrôlés par la température. Pendant la recherche, la température T ne diminue pas, d'où son influence directe sur la probabilité d'accepter une mauvaise solution. Cette probabilité tend vers 1 à une température élevée. Pour conclure, le rôle de la température T au cours du processus du recuit simulé est très important vu son influence sur l'algorithme [Creutz, 1983 ; Okamoto & Hansmann, 1995]. Le pseudo-code de l'algorithme du recuit simulé est illustré comme suit :

Algorithm 3 Recuit simulé

1: Initialiser (température T initiale)
2: Initialiser x (point de départ)
3: **Tant que** (non (fin)) **faire**
4: α=Voisin (x)
5: $\Delta E = E(\alpha) - E(x)$
6: **Si** $\Delta E < 0$ **alors**
7: $\alpha = x$
8: **Sinon** $alea(0,1) < \exp\left(\frac{-\Delta E}{T}\right)$ **alors**
9: $\alpha = x$
10: $T = \beta\,(T)$
11: **Si** $T < \Delta T$ **alors**
12: **Fin Tant que**
13: Répéter **Tant que**
14: **Fin Recuit simulé**

L'avantage du recuit simulé est qu'il peut offrir des solutions de bonne qualité, tout en restant simple à paramétrer et à programmer. Il présente aussi une souplesse d'emploi par rapport à l'algorithme de recherche locale. Parmi les inconvénients du recuit simulé figure celui-ci : une fois l'algorithme piégé à basse température, on aura un minimum local ; une autre difficulté est le choix de ses nombreux paramètres de contrôle.

1.5.2.3 Méthode de recherche tabou

La méthode de recherche Tabou est une technique de recherche proposée par Fred Glover en 1986.Elle est devenue classique en optimisation combinatoire, elle est classée comme méthode itérative de recherche locale à mémoire adaptative [Glover & Laguna, 1997 ; Glover, 1986 ; Glover, 1989a, 1989b ; Feo & Resende, 1989]. Réellement, la méthode de rechercheTabou est semblable àla méthode de recuit simulé.Elle fonctionne avec une seule configuration courante, qui est actualisée au cours des itérations.

Le principe de base de cette méthode consiste à explorer le voisinage d'une solution quelconque et à chercher le voisin qui minimisela fonction objectif. A chaque itération, on définit un voisinage pour une solution courante eton évalue l'objectif en chacune des solutions voisines. Selon le voisinage défini, cette opération peut entraîner une augmentation de la valeur de la fonction objectif à minimiser. Une solution voisine est adoptée même si elle est moins bonne. De ce fait, cette méthode permet d'éviter le piégeage dans les minimums locaux. Le risque de retourner à une solution déjà retenue lors d'une itération précédente peut se présenter. Pour éviter ce phénomène, l'utilisation d'une mémoire, conservant les informations sur les solutions déjà explorées, est retenue. Il s'agit de la mise à jour et de l'exploitation, à chaque itération, d'une liste des mouvements interdits, dite liste Tabou. Lorsqu'un minimum local est atteint, il y a interdiction de revenir sur le même chemin. La méthode Tabou fait qu'elle est largement employée pourdes problèmes d'optimisation combinatoire. Pour son efficacité, elle a été largement testée avec succès pour beaucoup de problèmes classiques, tels que les problèmes de voyageur de commerce, d'ordonnancement d'ateliers, etc. De même, elle est fréquemment appliquée sur les problèmes d'exploration géologique, de routage, d'ordonnancement, etc. La méthode de recherche Tabou peut être résumée par l'algorithme suivant [Glover, 1989b].

1.5.3 Métaheuristiques à population de solutions

Dans cette section, on présente les métaheuristiques à population de solutions contrairement aux algorithmes partant d'une seule solution. Les métaheuristiques de ce type améliorent une population de solutions au fur et à mesure des itérations. Plusieurs algorithmes ont déjà été conçus dans cette catégorie ; on peut citer les algorithmes évolutionnaires, qui font partie de la famille des algorithmes inspirés de la théorie de l'évolution de Charles Darwin [Darwin, 1859] et d'autres d'intelligence en essaim [Bonabeau et al., 1999].

Algorithm 4 Recherche Tabou

1: Déterminer une configuration aléatoire s
2: Initialiser une liste Tabou vide
3: **Tant que** le critère d'arrêt n'est pas satisfait **faire**
4: Perturbation de s suivant N mouvements non tabous
5: Évaluation des N voisins
6: Sélection du meilleur voisin t
7: Actualisation de la meilleure position connue s^*
8: Insertion du mouvement $t \leftarrow s$ dans la liste Tabou
9: $s = t$
10: **Fin Recherche Tabou**

1.5.3.1 Algorithmes d'intelligence en essaim

L'intelligence en essaim est une modélisation mathématique et informatique des phénomènes biologiques inspirés de l'éthologie [Bonabeau et al., 1999]. Ces algorithmes sont à base de population ; les agents ou les particules interagissent et coopèrent les uns avec les autres dans un environnement. Les interactions locales entre les agents conduisent souvent à l'émergence d'un comportement collectif global. Dans cette partie, on s'intéresse plus particulièrement aux algorithmes d'optimisation par essaim particulaire et de colonies de fourmis et à l'algorithme d'optimisation à base de biogéographie [Simon, 2008]. De nombreux algorithmes ont été développés dans plusieurs travaux comme l'optimisation par colonie d'abeilles [Walker et al., 1993], les systèmes immunitaires artificiels [Farmer et al., 1986] etc.

a. Algorithme à base de biogéographie

Cet algorithme, nommé BBO, a été conçu par Dan Simon en 2008 [Simon, 2008 ; Ma & Simon, 2011 ; Eegezer et al., 2009]. Il a pour origine la théorie de la biogéographie insulaire énoncée par MacArthur et Wilsonen 1967 [MacArthur & Wilson, 1967]. Cette approche de la biogéographie consiste en une répartition spatiale de différentes espèces, qui peuvent être animales ou végétales. Pour l'algorithme BBO, la population est formée d'habitats (îles). Chaque habitat peut être une solution d'un problème d'optimisation proposé. La "f*itness*" de chaque habitat est obtenue via la mesure de la qualité d'une solution appelée HSI (*Habitat Suitability Index*). Chaque habitat est désigné par des SIVs (*Suitability*

Index Variables). Pour cet algorithme, la meilleure solution sera une île qui contient le plus grand nombre d'individus, ce qui signifie un faible HSI. D'après les travaux de Wilson et MacArthur, le nombre d'individus d'une île est proportionnel aux taux d'immigration des nouveaux individus et aux taux d'émigration, nommée aussi extinction, des individus déjà installés sur l'île. Tous les habitats disposent d'un taux d'immigration λ et un autre d'émigration μ . Généralement, le taux d'immigration diminue avec l'augmentation du nombre d'individus, pour cela plus le nombre d'individus présents dans l'île augmente, plus d'autres immigrants appartenant à une nouvelle espèce rejoignent l'île et vice-versa. Ce taux devient maximal lorsque l'île est vide. Ce taux vaut zéro lorsque tous les individus sont présents sur l'île. On décrit cet algorithme BBO par le pseudo-code ci-dessous, dons lequel il est remarquable que ce sont les opérateurs de mutation et de migration qui en régissent le fonctionnement, le but étant de ne conserver que les meilleures solutions [Simon, 2008 ; Boussaid et al., 2012].

b. Optimisation par essaim particulaire

L'optimisation par essaim particulaire est une méthode d'optimisation métaheuristique inventée en 1995 aux Etats-Unis par Kennedy et Eberhart [Kennedy & Eberhart, 1995a, 1995b ; Shi & Eberhart, 1999]. L'appellation d'origine étant, en anglais, Particle Swarm Optimisation (PSO).

Cette technique est inspirée du comportement social des animaux (éthologie) dans le but de simuler des interactions sociales. En outre, Kennedy et Eberhart ont essayé, dès le début, de modéliser les interactions entre des "*agents*" dont l'objectif est d'atteindre le résultat cherché dans un espace de recherche commun. Pour atteindre cet objectif, la coopération et l'échange d'informations entre les agents sont des critères indispensables [Borne & Benrejeb, 2010]. Dans ce cadre, on distingue quelques notions essentielles pour cette métaheuristique : positions, vitesses, échange d'informations, mémoire et capacité de combiner les informations pour la décision. Chaque particule essaye de bouger, c'est-à-dire qu'elle a une vitesse, également chaque particule a une mémoire qui lui permet de se souvenir de sa meilleure performance en positions et en valeurs. Enfin, chaque particule dispose d'un groupe d'informatrices, historiquement appelées voisinage [Clerc & Siarry, 2009 ; Eberhart & Shi, 2001a]. Le déplacement d'une particule est influencé par les trois composantes suivantes :

- une composante physique : la particule tend à suivre sa direction courante de déplacement,

Algorithm 5 BBO

1: Générer aléatoirement un ensemble d'îles (solutions initiales)
2: **Tant que** le critère d'arrêt n'est pas atteint **faire**
3: Évaluer la *fitness* (HSI) de chaqueindividu(solution)
4: Calculer le nombre d'espèce *nbesp*, le taux d'immigration λ et
5: d'émigration μ pour chaque individu
6: **Migration :**
7: **Pour** $i = 1$ à N **faire**
8: Utiliser λ_i pour décider, de manière probabiliste, d'immigrer à x_i
9: **Si** $rand(0, 1) \leq \lambda_i$ **alors**
10: **Pour** $j = 1$ à N **faire**
11: Sélectionner l'île d'émigration x_j
12: **Si** $rand(0, 1) \leq \mu_j$ **alors**
13: Remplacer une variable de décision (SIV) choisie aléatoirement dans x_i
14: par la variable correspondante dans x_j
15: **Fin Si**
16: **Fin Pour**
17: **Fin Si**
18: **Fin Pour**
19: **Mutation :**
20: Muter les individus au taux de mutation
21: Remplacement de la population par les descendants
22: Implémenter l'élitisme
23: **Fin Tant que**
24: Retourner la meilleure solution trouvée
25: **Fin BOO**

- une composante cognitive : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée,
- une composante sociale : la particule tend à se fier à l'expérience de ses congénères et, ainsi, à se diriger vers le meilleur site déjà atteint par ses voisins.

La figure 1.14 illustre la stratégie de déplacement d'une particule.

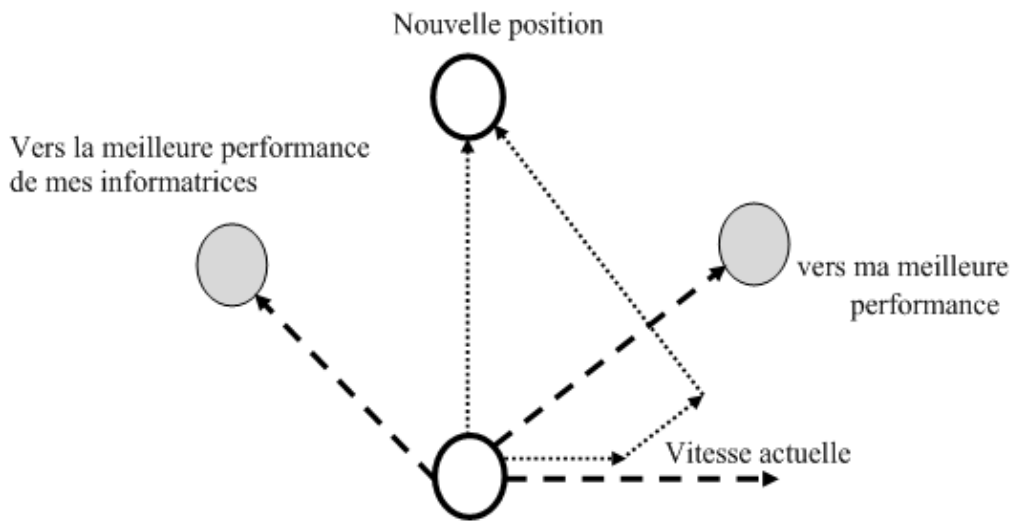

FIGURE 1.14 – Déplacement d'une particule dans l'essaim.

c. Algorithme de colonies de fourmis

L'optimisation par colonie de fourmis ACO a été proposée par Marco Dorigo et ses collègues en 1992, afin de résoudre le problème du voyageur de commerce ayant pour objectif de visiter un certain nombre de villes, en minimisant la distance parcourue [Dorigo et al., 1991, 1996].

Tout problème d'optimisation considéré dans ce cadre est modélisé comme la recherche d'un chemin de coût minimum. Ce concept est inspiré du comportement des fourmis lors de la recherche du plus court chemin reliant la fourmilière à une source de nourriture, en s'adaptant aux éventuels changements de l'environnement, Figure 1.15. Le principe de cette métaheuristique est décrit comme suit. Pour communiquer, chaque fourmi dépose, le long de son chemin, une substance chimique volatile, appelée phéromone, et marque ainsi son trajet. Celle qui trouve la source de nourriture en premier, revient en suivant sa propre trace, déposant à nouveau de la phéromone [Goss et al., 1989 ; Colorni et al., 1991 ; Lopez-Ibanez et al., 2015]. Cette méthode exploite ces caractéristiques pour construire un vecteur de solutions pour un problème d'optimisation. Le phénomène d'évaporation sert à explorer l'espace de recherche à chaque itération. L'algorithme serait piégé dans un optimum local sans ce phénomène. Bien que conçu au départ pour le problème du voyageur de commerce, l'algorithme de colonies de fourmi offre finalement beaucoup de souplesse. Il a été adapté à plusieurs problèmes combinatoires. Par exemple, pour le problème des

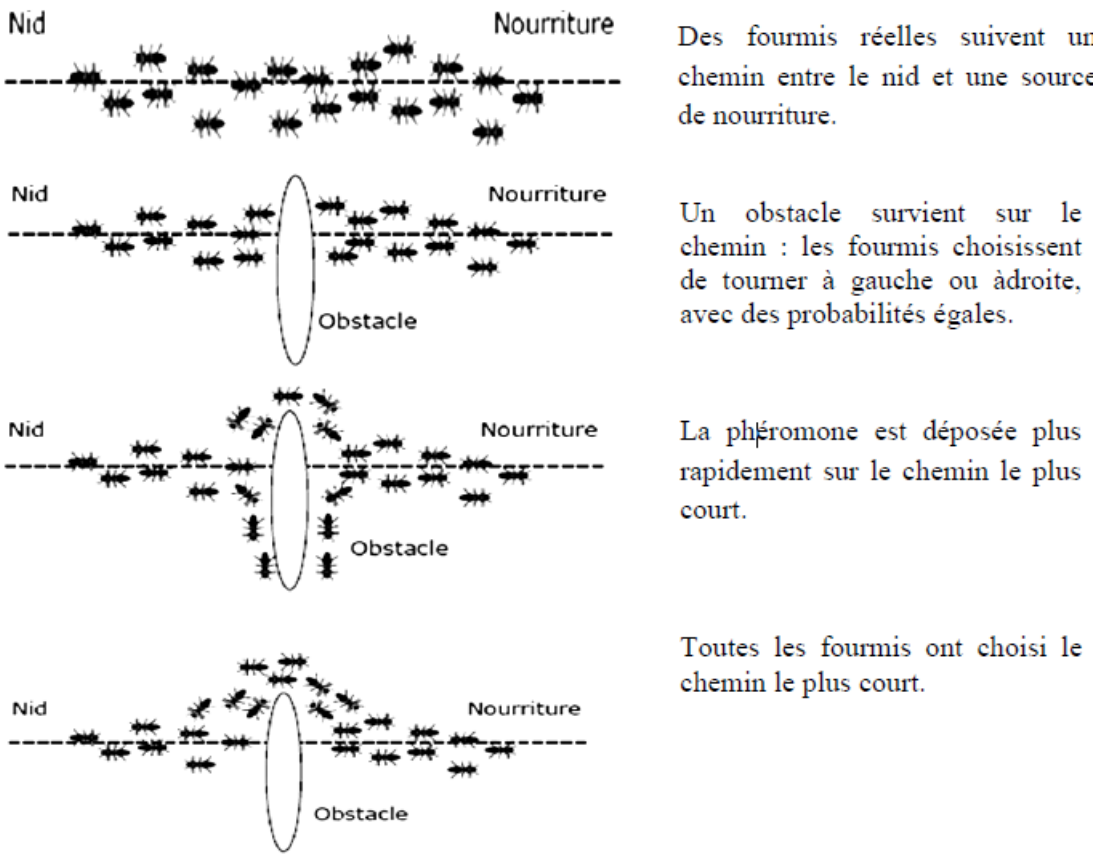

FIGURE 1.15 – Faculté d'une colonie de fourmis à retrouver le plus court chemin, fortuitement obstrué par un obstacle [Tfaili, 2012].

tournées de véhicules, on peut utiliser deux colonies de fourmis simultanément, chacune gérant ses propres pistes de phéromone : une colonie minimise le nombre de véhicules utilisés, et une autre minimise le coût total des tournées. L'algorithme, de par son dynamisme intrinsèque, s'adapte aussi facilement aux espaces de solutions qui varient dynamiquement dans le temps.

1.5.3.2 Algorithmes évolutionnaires

Les algorithmes évolutionnaires sont des algorithmes basés sur la théorie de l'évolution et la sélection naturelle, ils ont été proposés à l'origine par Darwin en 1985 pour résoudre plusieurs problèmes d'optimisation [Fogel et al., 1966 ; Fraser, 1960]. Le principe fondamental des algorithmes évolutionnaires est d'adapter les individus à leur lieu ou environnement pour survivre et se reproduire en transmettant leurs gènes aux descendants. En effet, ils

permettent de faire évoluer une population pour la recherche des solutions dans un environnement. Toutes les solutions trouvées, appelées aussi individus, sont représentées par leur génotype, exprimé sous la forme d'un phénotype.

Plusieurs métaheuristiques sont inspirées du terme Evolutionary Computation, telles que les stratégies d'évolution [Rechenberg, 1965 ; Beyer, 2001], la programmation évolutive [Fogel et al., 1966 ; Fogel, 1962], les algorithmes génétiques [Holland, 1975], l'évolution différentielle [Storn & Price, 1997 ; Price et al., 2005] et la programmation génétique [Koza, 1992]. La figure 1.16 illustre le principe des algorithmes évolutionnaires basés essentiellement sur les opérateurs de sélection, de croisement et de mutation.

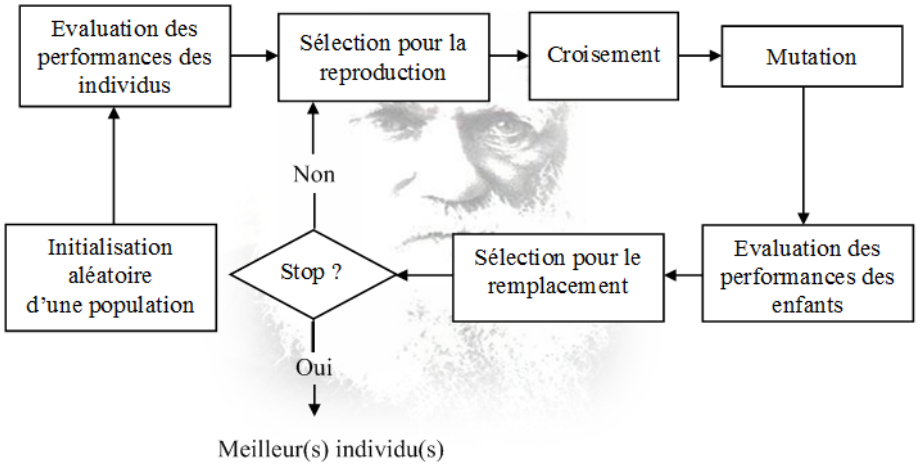

FIGURE 1.16 – Principe d'un algorithme évolutionnaire standard [Coello.Coello et al., 2007].

Comme les algorithmes évolutionnaires sont itératifs, chaque itération correspond à une génération. La population est l'ensemble d'individus qui représentent les solutions prêtes pour le croisement et capables de se reproduire. La phase d'évaluation consiste à calculer la fitness pour chaque individu par rapport à son phénotype. La phase de reproduction englobe deux enchaînements ; on croise les individus de la population (génotypes) selon un opérateur de croisement. On obtient donc des nouveaux individus, appelés aussi descendants ou enfants. Ensuite, certains descendants vont muter, ce qui signifie la modification aléatoire d'une partie de leur génotype. La dernière étape consiste à sélectionner une partie des descendants pour former une nouvelle population de la même taille que la population initiale. L'opérateur de sélection ne prendra par exemple que les meilleurs individus en fonction de leur fitness.

Vu que les algorithmes évolutionnaires permettent la mise en place de stratégies d'optimisation efficaces, on réserve la suite du chapitre à l'étude détaillée de l'algorithme génétique et de l'algorithme d'évolution différentielle, surtout dans un formalisme d'optimisation multiobjectif, axe central de ces travaux de recherche.

a. Algorithmes génétiques

L'algorithme génétique GA a été développé en 1975 par J. Holland [Holland, 1975 ; Borne & Benrejeb, 2010 ; Dréo et al., 2003 ; Goldberg, 1994]. Cet algorithme est le plus populaire et le plus largement utilisé des algorithmes évolutionnaires. Il est inspiré de l'évolution biologique des espèces vivantes, mettant en oeuvre les phénomènes de sélection naturelle et de reproduction issus de la théorie de Charles Darwin.

L'ensemble des individus forment la population des parents, qui va évoluer au cours des itérations, dites "générations", jusqu'à ce qu'un critère d'arrêt soit vérifié. Cet algorithme est basé sur les opérateurs génétiques de sélection, de croisement et de mutation, décrits dans ce qui suit :

- Sélection : pour passer d'une génération à une autre, nous soumettrons la population à des opérateurs de sélection. Les opérateurs de variation, eux, permettront de transformer la population de façon à favoriser l'émergence de meilleurs individus. L'algorithme génétique repose sur une boucle qui enchaîne des étapes de sélection et des étapes de croisement. Dans un premier temps, à partir d'une population d'individus, on désigne ceux autorisés à se reproduire. Les techniques de sélection les plus largement utilisées sont la sélection par tirage à la roulette (*roulette-wheel selection*), la sélection par rang (*ranking selection*) et la sélection par tournoi (*tournament selection*) [Goldberg & Deb, 1991 ; Blickle & Thiele, 1995].
- Croisement : on croise ensuite ces individus de façon à obtenir une population d'enfants. Comme l'étape de sélection, on distingue plusieurs techniques de croisement, on peut citer le croisement uniforme et le croisement en un point.
- Mutation : on peut faire muter aléatoirement certains gènes. La performance des enfants est évaluée grâce à la fonction fitness. La procédure est répétée, et l'on recommence une phase de sélection pour la reproduction et une phase de mutation, et ainsi de suite [Goldberg, 1994].

Comme pour les métaheuristiques vues précédemment, un critère d'arrêt permet de sortir de la boucle. Ceci peut se traduire par un certain nombre d'itérations sans amélioration

notable de la performance des individus. Les algorithmes génétiques sont coûteux en temps de calcul car ils manipulent plusieurs solutions simultanément. Parmi les avantages des algorithmes génétiques, on cite leur capacité à trouver de bonnes solutions pour des problèmes d'optimisation très complexes.

b. Évolution différentielle

L'algorithme à évolution différentielle DE a été proposé par R. Storn et K. Price en 1997 pour résoudre le problème des polynômes de Tchebychev. Cette méthode est classifiée parmi les métaheuristiques stochastiques d'optimisation [Lampinen & Zelinka, 1999]. Les concepts de base de l'évolution différentielle sont inspirés de la combinaison des algorithmes génétiques et des stratégies évolutionnaires, ainsi que de la technique géométrique de recherche. Contrairement aux algorithmes génétiques, qui modifient la structure des individus par les phases de mutation et croisement, l'algorithme à évolution différentielle est basé sur une auto-adaptation par une manipulation géométrique des individus. Price et al. [Price et al., 2005] ont défini les variantes de l'algorithme en question par $DE/x/y/z$ dont x est le mode de sélection des individus cibles ou de référence, nommés respectivement par **best** et **rand** pour la phase de la mutation, y présente le nombre de différenciations et z correspond aux schémas de mutation, qui peuvent être exponentiel ou binomial. Au cours des itérations et en se basant sur les stratégies de mutation, chaque individu est généré en ajoutant à ses composants la différence pondérée d'autres individus de la population. Puis, un vecteur d'essai est généré suite à la phase de croisement entre l'individu initial et son vecteur dit mutant. La phase de sélection consiste à comparer les valeurs de la fonction objectif de deux vecteurs père et descendant, pour garder le vecteur ayant la plus petite valeur de fonction objectif, en cas de minimisation. Ce processus conduit à la création d'une nouvelle population. Il sera répété jusqu'à atteindre le nombre maximaldéfini de générations [Storn & Price, 1997 ; Price et al., 2005 ; Xiaobing et al., 2015 ; Zhaoa et al., 2016].

1.6 Optimisation multiobjectif de lois de commande robuste RST : position du problème

Plusieurs difficultés sont rencontrées en automatique, plus précisément, la synthèse de la commande polynômiale RST. L'inconvénient majeur de la technique de placement de pôles est toujours le choix des pôles en boucle fermée, qui est généralement délicat et de-

vient plus difficile avec la complexité du système à commander. Jusqu'à maintenant, il n'y a pas de procédure claire et systématique pour guider le choix des pôles. Pour surmonter ce problème, plusieurs techniques ont été proposées dans la littérature. Les deux méthodes classiques, sur la base de placement de pôles avec calibrage des fonctions de sensibilité, ont été proposées par Landau et Karimi [Landau & Karimi, 1998]. La première approche combine le placement de pôles et la calibration des fonctions de sensibilité, en utilisant des parties fixes dans le contrôleur. Elle ajuste itérativement les fonctions de sensibilité dans le domaine fréquentiel. La deuxième méthode est proposée dans [Landau & Karimi, 1998], il s'agit de définir et calculer une fonction de sensibilité de sortie désirée en satisfaisant des contraintes de robustesse, de type $\mathcal{H}_\infty$, des filtres de pondération associés au processus à commander. L'utilisation d'une partie fixe appropriée de la fonction de sensibilité de sortie désirée assurera un placement adéquat des pôles auxiliaires du régulateur RST. L'inverse de cette fonction de sensibilité désirée définit le filtre de pondération $\mathcal{H}_\infty$ relatif. Dans cette approche, la sélection du filtre de pondération est réalisée automatiquement par un programme d'optimisation. Rotella et al. [Rotella et al., 2001] ont proposé une nouvelle approche pour la synthèse de régulateurs numériques RST basée sur la propriété de platitude des systèmes dynamiques.

La complexité relevant de la synthèse peut être mieux gérée en cherchant sa formulation comme un problème d'optimisation multiobjectif. Notre orientation pour cette thèse est de proposer des formulations pour le problème de synthèse RST dans un cadre multiobjectif. Vu la complexité de ce genre de problème, cette orientation est justifiée par le fait d'optimiser plusieurs critères généralement contradictoires. Ceci mène à plus de performance au niveau des réponses de systèmes dynamiques et plus de robustesse. Le point crucial est de définir des critères à optimiser, puisque cela constitue une phase très délicate, car intrinsèquement plusieurs paramètres entrent en jeu. L'ajustement systématique des paramètres des régulateurs pour des systèmes complexes constitue un problème souvent délicat lors de l'élaboration d'une structure de commande. En effet, notre travail entre ainsi dans ce cadre et tente de développer une approche systématique pour la synthèse de la commande polynômiale RST. Les problèmes d'optimisation multicritère formulés seront résolus en utilisant une technique d'optimisation multiobjectif basée sur l'approche des essaims particulaires, avec des perfectionnements variés au niveau de l'algorithme de base.

1.7 Conclusion

Dans ce chapitre, nous avons présenté, en premier lieu, des généralités sur la théorie de la commande robuste. Les principales approches de synthèse relevant de ce cadre ont été évoquées dans le but de montrer la complexité de leur résolution par des techniques classiques. Les problèmes de synthèse et de réglage des structures de commande RST des systèmes complexes ont été plus particulièrement soulignés. Pour gérer la complexité, l'idée de poser le problème de synthèse sous forme d'un problème d'optimisation, où plusieurs techniques métaheuristiques pouvant être utilisées, a été proposée. Nous avons clôturé ce chapitre par la présentation de quelques métaheuristiques de voisinage et d'autres à base de population, plus particulièrement la méthode des essaims particulaires PSO, en vue d'en choisir les plus adaptées aux problèmes de commande polynômiale RST, retenu dans un formalisme multiobjectif comme problématique au centre de ces travaux de recherche.

Le chapitre suivant présentera les concepts de base des algorithmes d'optimisation multiobjectif et plusieurs algorithmes évolutionnaires, plus particulièrement les deux algorithmes développés : NSGA-II et MODE.

Chapitre 2

Algorithmes évolutionnaires d'optimisation multi-objectif

2.1 Introduction

Les problèmes à objectifs multiples se posent et surgissent de façon naturelle dans la plupart des disciplines. Leur résolution a été un défi pour les chercheurs depuis longtemps [Coello.Coello et al., 2007 ; Deb & Jain, 2013 ; Zhao et al., 2012]. Malgré la variété considérable des techniques développées en recherche opérationnelle et d'autres disciplines pour résoudre ces problèmes, la complexité de leurs solutions exige des approches alternatives.

L'utilisation des algorithmes évolutionnaires (EAs) pour résoudre les problèmes d'optimisation multi-objectif, alias MOP pour Multi-Objective Problems, a été motivée principalement en raison de la nature des EAs, basée sur la population. En outre, la complexité de certains MOP (grandes dimensions d'espaces de recherche, incertitude, contraintes, etc.) peut empêcher l'utilisation des techniques traditionnelles telles que l'OR-MOP. Les MOP sont manipulés aujourd'hui par des ingénieurs, des informaticiens, des biologistes et des chercheurs en utilisant des algorithmes évolutionnaires.

Ce chapitre présentera un aperçu général sur l'optimisation multi-objectif évolutionnaire (EMO), qui se réfère à l'utilisation des algorithmes évolutionnaires pour la résolution de problèmes de commande multicritères considérés dans nos travaux. Parmi des plus utilisés, on a choisi l'algorithme génétique NSGA-II, ainsi que celui d'évolution différentielle MODE pour résoudre la problématique abordée dans ces travaux de recherche.

2.2 Optimisation multi-objectif difficile

Dans cette section, des notions et concepts de base relatifs à l'optimisation multi-objectif seront présentés. Ces notions présentent les bases des approches proposées et développées pour résoudre le problème de synthèse RST multi-objectif évoqué dans le premier chapitre et reformulé ultérieurement.

2.2.1 Notion de problème d'optimisation multi-objectif contraint

Les problèmes d'optimisation multicritères ont pour objectif de satisfaire les besoins et les spécifications, le plus souvent contradictoires, pour mieux répondre aux exigences du cahier des charges [Van-Veldhuizen, 1999 ; Zitzler et al., 2010 ; Collette & Siarry, 2004 ; Coello.Coello et al., 2007 ; Dréo et al., 2003].

La principale difficulté rencontrée en optimisation monobjectif, et qui peut être une tâche difficile, est la formulation du problème sous la forme d'une équation unique. L'optimisation multi-objectif autorise ces degrés de liberté qui manquaient en optimisation monobjectif. Comme pour les problèmes d'optimisation monobjectif, le but de l'optimisation multi-objectif est de chercher l'optimum d'une fonction coût qui peut être un minimum ou un maximum. Dans la plupart des problèmes d'optimisation, il y a toujours des restrictions imposées par les caractéristiques particulières de l'environnement ou des ressources disponibles (limitations physiques, restrictions de temps, etc.).

Ces restrictions doivent être respectées afin d'examiner les solutions acceptables. En général, toutes ces restrictions sont appelées contraintes, décrivant les dépendances entre les variables de décision et les constantes (ou paramètres) impliqués dans le problème. Par conséquent, ces problèmes peuvent être posés par la présence des contraintes c'est-à-dire que les variables de la fonction objectif à optimiser sont contraintes d'évoluer dans une partie de l'espace de recherche du problème considéré [Coello.Coello et al., 2007 ; Dréo et al., 2003 ; Collette et al., 2002].

L'image de l'espace de recherche, nommé aussi espace réalisable par la fonction objectif f, est appelée espace des objectifs ou espace des critères. L'évaluation des fonctions MOP $f : \Omega \rightarrow \Lambda$ transforme les variables de décisions $x = (x_1, x_2, ..., x_m)$ aux vecteurs $y = (a_1, a_2, ..., a_k)$. Cette situation est représentée à la figure 2.1 pour le cas $m = 2$ et $k = 3$. Dans le paragraphe suivant, on présentera la notion de la dominance qui est le point d'articulation dans le formalisme de l'optimisation multi-objectif. Nous classifierons aussi les approches de résolution des MOP en trois catégories différentes : une approche

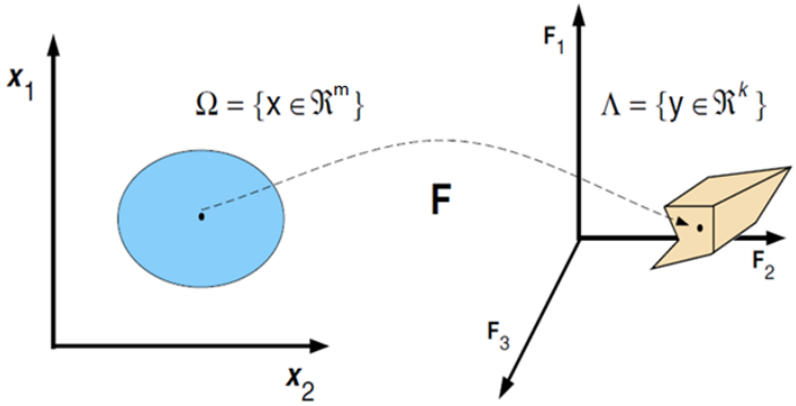

FIGURE 2.1 – Représentation de l'espace de recherche et de son image par la fonction multi-objectif.

dite agrégative, basée sur la transformation du problème d'optimisation en un autre monobjectif, une approche Pareto et approche non Pareto.

2.2.2 Notion de dominance

La résolution d'un problème d'optimisation multi-objectif conduit à l'obtention d'une multitude de solutions et seul un nombre réduit de ces solutions va être retenu. Ce choix délicat des solutions est basé sur la relation de dominance. Cette relation existe entre la solution considérée et les autres solutions. Le but de l'optimisation monobjectif est de trouver un optimum global présentant toujours la solution unique à retenir. Au contraire, l'optimisation multi-objectif fournit une surface des solutions qui est un compromis entre toutes les fonctions à optimiser.

Ainsi, la notion d'optimalité dans le formalisme multi-objectif est basée sur ce principe de dominance. Pour cela on introduit les définitions suivantes [Coello.Coello et al., 2007 ; Deb, 1999 ; Van-Veldhuizen , 1999 ; Bagchi, 2001 ; Bäck, 1996] :

Définition 2.1 : Étant donné x, $y \in \Re^m$, on dit que $x \leq y$ si $x_i \leq y_i$ pour $i = 1, \ldots, m$, et que x **domine** y (noté $x \prec y$) si $x \leq y$ et $x \neq y$.

Définition 2.2 : On dit qu'un vecteur $x \in \Omega \subset \Re^m$, solution du problème, est **non-dominé** dans Ω s'il n'existe pas d'autre vecteur $x' \in \Omega$ tel que $f(x') \prec f(x)$.

La figure 2.2 illustre le concept de dominance dans le cas d'un problème d'optimisation bi-objectif. Les solutions représentées par les points en couleur bleue ne sont pas dominées

par aucune autre solution.

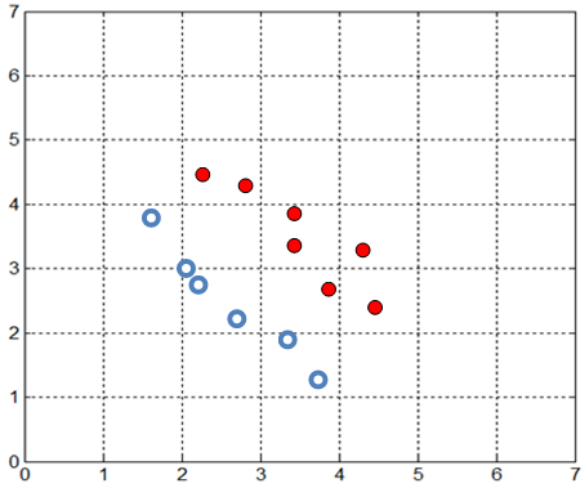

FIGURE 2.2 – Illustration de la notion de dominance pour un problème d'optimisation bi-objectif.

2.2.3 Approche agrégative

Cette approche de résolution consiste à transformer un problème multicritère en un autre mono-critère. A l'aide de l'équation (2.1), on obtient une "combinaison" de toutes les fonctions f_i à optimiser. La forme typique de la fonction d'agrégation linéaire est donnée par :

$$fitness = \min \sum_{i=1}^{M} w_i f_i(x) \tag{2.1}$$

avec $w_i \geq 0$ et $\sum_{i=1}^{M} w_i = 1$ sont les coefficients de pondération relatifs à M fonctions objectifs du MOP.

Cette approche ne produit qu'une seule solution. Elle est articulée sur deux points essentiels ; comment déterminer les paramètres de pondération associés à chaque critère et quelle est l'interaction entre les différents critères [Collette & Siarry, 2004 ; Coello.Coello et al, 2007 ; Berro, 2001]. La figure 2.3 illustre le principe de fonctionnement de cette technique dont le calcul des valeurs de pondération revient à trouver une partie de l'espace de recherche dite hyper-plan. Comme le montre la figure 2.3-a, la solution réalisable sera le point où l'hyper-plan a une tangente commune avec l'espace des solutions réalisables. Le point x présente une solution non-dominée. Dans le cas d'un front de Pareto non convexe

(concave), cette approche ne peut trouver que les solutions y et z et risque de ne pas retenir toutes les solutions entre ces variables, Figure 2.3-b. Ceci présente l'inconvénient majeur de cette approche.

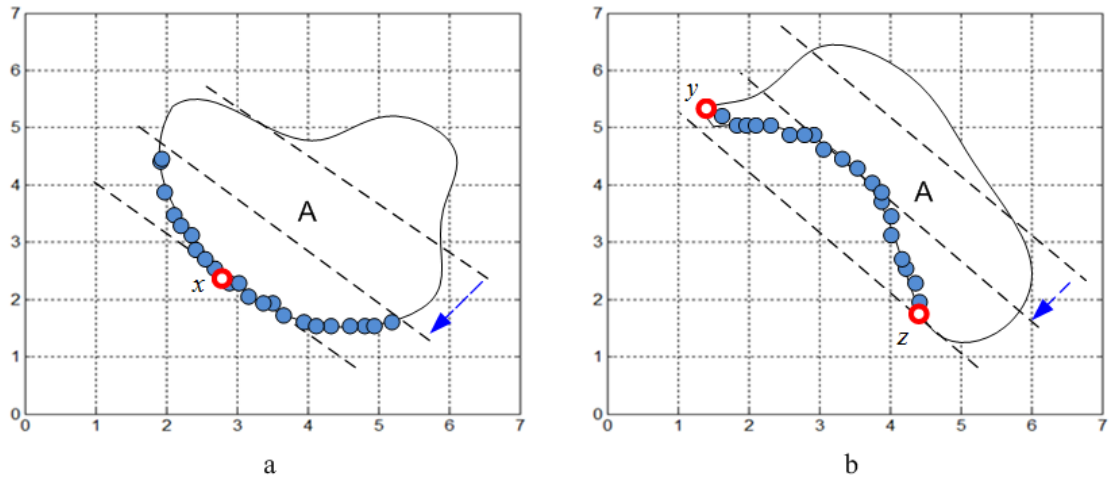

FIGURE 2.3 – Illustration de la méthode d'agrégation en cas d'optimisation bi-objectif : (a) Une frontière de Pareto convexe, (b) Une frontière de Pareto non convexe. A : domaine réalisable.

2.2.4 Approche non Pareto

Cette approche est classifiée comme une approche non agrégée. Généralement, les techniques d'optimisation basées sur cette stratégie traitent les objectifs séparément. Dans la littérature, il existe deux techniques non Pareto [Schaffer, 1985] : la sélection parallèle et la sélection lexicographique.

La première approche, connue grâce aux travaux de Schaffer [Schaffer, 1985 ; Collette, 2002], est basée sur l'algorithme génétique nommé VEGA. La deuxième technique est réalisée suivant un ordre défini a priori. Cet ordre permet de définir les poids des objectifs. Plusieurs métaheuristiques ont été utilisées pour la résolution des MOPs avec la sélection lexicographique [Jungjit & Freitas, 2015].

2.2.5 Approche Pareto

Cette approche est basée sur la notion de dominance, décrite dans la section précédente. Plusieurs algorithmes l'utilisent pour produire une frontière, le front de Pareto, nommé

aussi surface de compromis, qui contient toutes les solutions non dominées du problème. Conçue par Goldberg en 1989 [Goldberg et al., 1989], cette approche provoque un calcul équitable des solutions pour touts les cirières à optimiser. Ayant plusieurs fonctions objectifs, la notion de l'optimum change car pour les MOPs l'objectif est de trouver les bons compromis au lieu d'une seule solution, figure 2.4. Cette approche prouve son efficacité dans plusieurs travaux et avec plusieurs algorithmes d'optimisations multi-objectifs [Collette et al. 2004 ; Coello.Coello et al, 2007 ; Coello.Coello, 2002 ; Bagchi, 2001 ; Schaffer, 1985]. Pour un tel problème d'optimisation, la solution $x^* \in \Re^m$ est appelée une solution

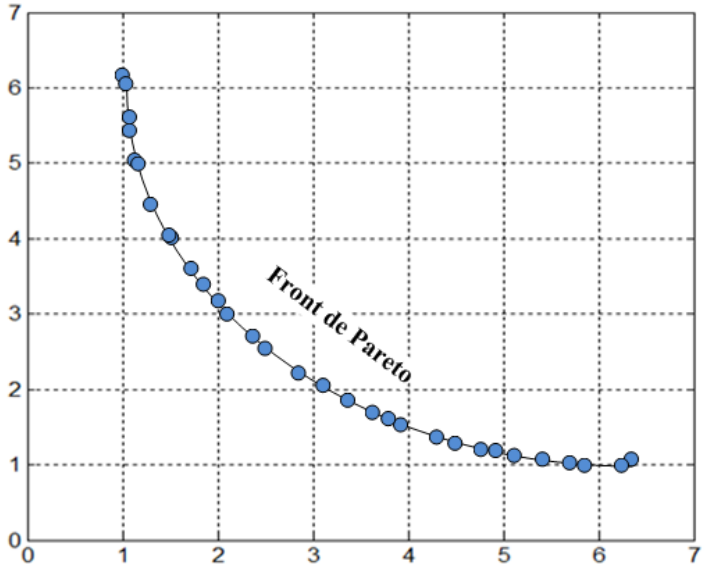

FIGURE 2.4 – Front de Pareto dans le cas d'optimisation bi-objectif.

optimale au sens de Pareto, s'il n'y a pas de variables de décision $x \in \Re^m$ satisfaisant $f_i(x) \leq f_i(x^*)$, $(i = 1, 2, \ldots, M)$. En d'autres termes, x^* est un optimum de Pareto, s'il n'existe aucun vecteur possible qui diminuerait certains objectifs sans entraîner une augmentation dans au moins un autre objectif simultanément. Cela revient à dire que les solutions optimales sont des solutions qui ne sont pas dominées par d'autres solutions. Contrairement à l'optimisation monobjectif ayant une seule solution, l'optimisation multi-objectif mènera à un ensemble de solutions, appelé l'ensemble de Pareto optimal, noté P* . L'image de cet ensemble de Pareto dans l'espace objectif est appelée le front de Pareto. En utilisant les notations ci-dessus, le front de Pareto et le Pareto optimal peuvent être définis comme suit :

Définition 2.3 : L'ensemble optimal de Pareto est défini par :

$$\mathrm{P} = \{x \in \Omega | \neg \exists\, x' \in \Omega \; f(x') \, f(x) \;\} \tag{2.2}$$

Définition 2.4 : Le front optimal de Pareto est défini par :

$$P^* = \left\{ f(x) \in \Re^M | x \in \mathrm{P}^* \right\} \tag{2.3}$$

2.3 Concepts de base des algorithmes évolutionnaires

Dans ce paragraphe, on définit les termes les caractérisant la structure des algorithmes évolutionnaires qui sont analogues à leurs homologues génétiques [Coello.Coello et al., 2007 ; Deb, 1999 ; Coello.Coello, 2001 ; Deb & Jain, 2013 ; Zhao et al., 2012]. Ces concepts sont illustrés dans la figure 2.5. Une structure ou un individu est une solution codée à un

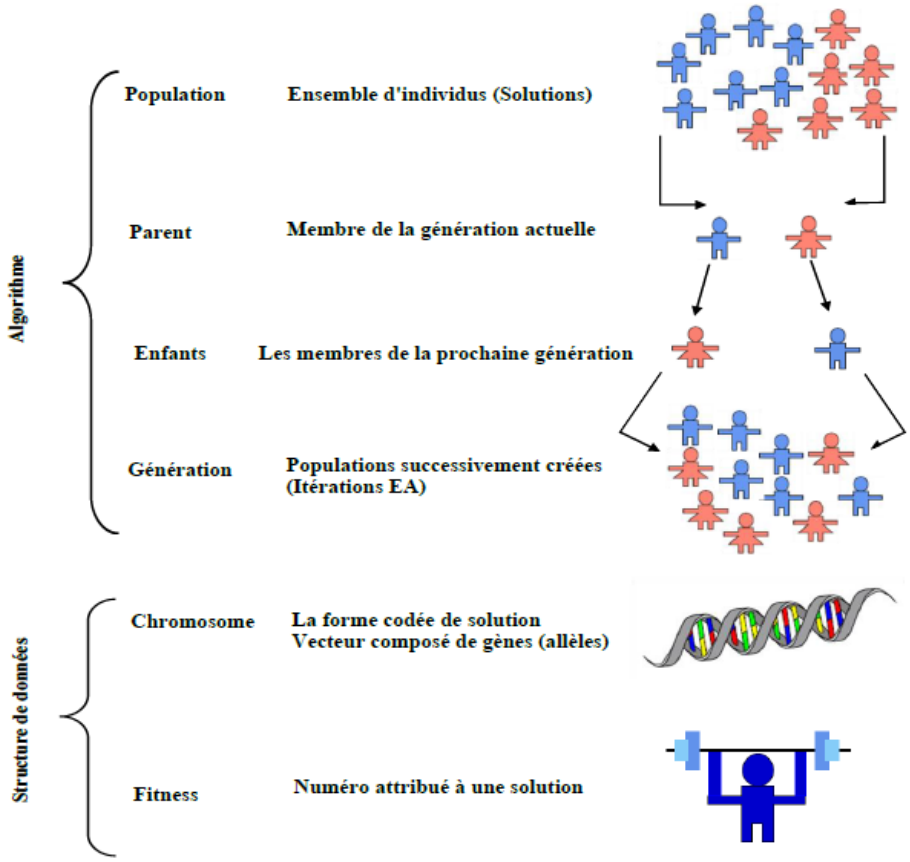

FIGURE 2.5 – Composants clés de l'EA [Coello.Coello et al., 2007].

problème. Typiquement, un individu est représenté comme une chaîne de code similaire à

un génotype biologique. Ce génotype définit un organisme individuel lorsqu'il est exprimé (décodé) en un phénotype. Un génotype est composé d'un ou plusieurs chromosomes, où chacun est composé de gènes distincts. Ainsi, chaque individu décodé est un ensemble de paramètres utilisés comme entrée de la fonction objectif. Finalement, un ensemble donné de chromosomes est appelé une population. Tout comme dans la nature, les opérateurs évolutionnaires (EVOPs) manipulent une population d'individus et tentent de générer des solutions. Les trois EVOPs de base pour les algorithmes évolutionnaires (EAs) sont : la mutation, le croisement (recombinaison) et la sélection. La figure 2.6 présente une mutation au niveau d'un bit sur une chaîne codée, où le bit "1" est changé par un "0", ou vice versa. La figure 2.7 montre le croisement d'un seul point (une forme de recombinaison) pour deux chaînes binaires (parents). Chaque parent est coupé et recombiné avec une pièce de l'autre [Goldberg, 1994]. Les meilleurs individus dans la population sont sélectionnés

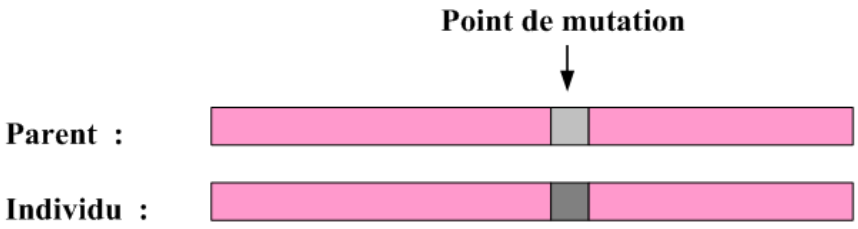

FIGURE 2.6 – Mutation au niveau d'un bit.

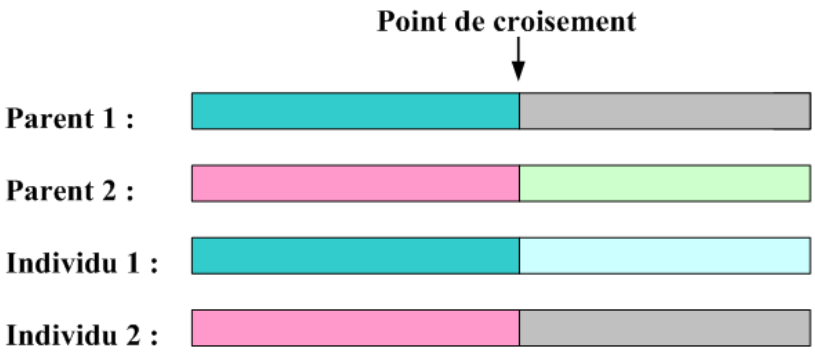

FIGURE 2.7 – Croisement en un seul point.

pour devenir membres de la prochaine génération. Les valeurs réelles des chromosomes subissent également les mêmes opérateurs évolutionnaires (EVOPs). Il existe de nombreuses variations sur les opérateurs de base des algorithmes évolutionnaires qui dépendent des

contraintes du problème posé. Ces contraintes peuvent influencer la structure des chromosomes et des allèles. Il existe plusieurs méthodes de sélection, telles que la sélection par roulette et la sélection par tournoi [Bäck, 1996].

2.4 Algorithmes génétiques multi-objectif

2.4.1 Principe de base d'un algorithme génétique mono-objectif

Dans cette partie, on présente les principes de fonctionnement et les éléments de base caractérisant un algorithme génétique (AG). On s'intéresse dans ces travaux de recherche aux algorithmes génétiques multi-objectifs qui sont basés sur les concepts d'un AG standard. D'après Coello [Coello.Coello, 2002], ce dernier est classé parmi les algorithmes stochastiques d'optimisation. Cet algorithme est caractérisé par une population d'individus (ou de solutions) et est basé sur le codage des paramètres. En plus, il utilise des règles de transition probabiliste et non déterministe. L'algorithme génétique n'utilise que les valeurs de la fonction à optimiser et non pas sa dérivée, ou une autre connaissance auxiliaire. Ainsi, un AG standard exige le codage de l'ensemble des paramètres du problème à optimiser en une chaine de longueur finie. La simulation d'un AG provoque l'évolution d'une population initiale d'individus. Au début, une procédure de génération de la population aura lieu d'une façon aléatoire. Ensuite, les individus seront sélectionnés avec une fonction d'adaptation. Ceci mène à la création d'une nouvelle population "Enfant". Par la suite, les opérations de croisement et de mutation sont appliquées à cette dernière. Un critère d'arrêt sera mis toujours en considération pour la fin des générations.

L'AG standard est classé parmi les algorithmes générationnels car la population "Enfant" généré va remplacer la population "Parent". La phase la plus importante en termes de qualité des solutions est la sélection d'individus à retenir pour les prochaines générations. La sélection est une procédure qui mène à filtrer les individus qui doivent se reproduire pour avoir des meilleurs individus. L'identification de ces derniers est basée sur la fonction d'adaptation. Les moins bons d'individus seront écartés et les restants dans la population ont une valeur de probabilité élevée ce qui implique leur contribution à la génération suivante. Finalement, l'organigramme de la figure 2.8 résume les grandes lignes d'un algorithme génétique standard [Holland, 1975 ; Goldberg, 1994].

Pour les algorithmes génétiques, la technique de sélection par tournoi consiste à choisir aléatoirement n individus de la population sans tenir compte de la valeur de la fonction

d'adaptation, et choisir les meilleurs parmi eux. La deuxième technique, en l'occurrence la $I_{\max}$ occupe une place qui est proportionnelle à la valeur de sa fonction d'adaptation $fitness\,(I)$. La probabilité de sélection est définie par l'équation (2.4) :

$$Prob\,(I) = \frac{fitness\,(I)}{\sum_{I=1}^{I_{\max}} fitness\,(I)} \tag{2.4}$$

A chaque sélection, un tirage à la "loterie" aura lieu. Généralement, les individus qui ont

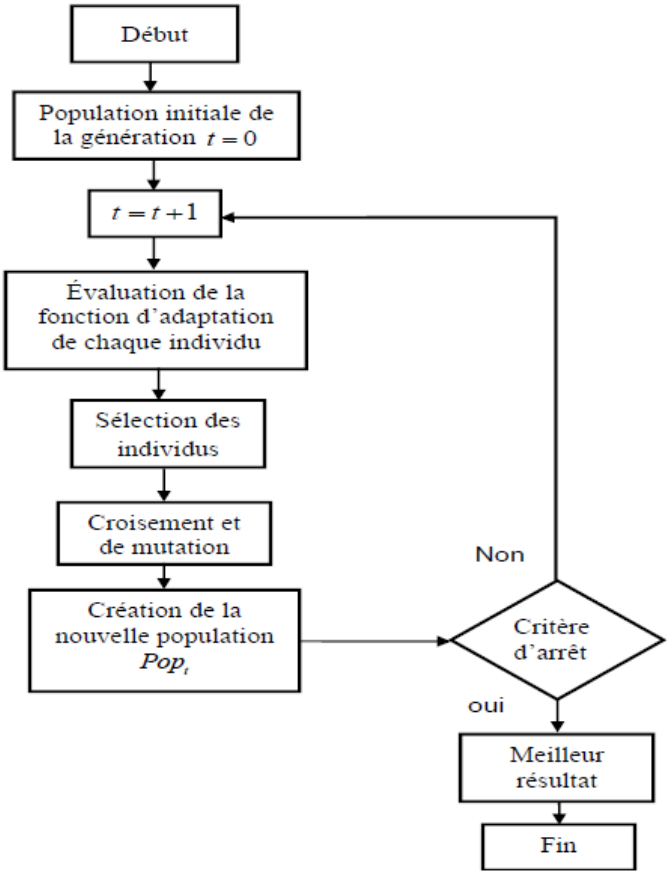

FIGURE 2.8 – Organigramme d'un AG standard.

les plus grandes valeurs de $fitness\,(I)$ ont plus de chance pour être choisis. Le croisement des individus permet de créer des nouvelles chaînes en échangeant des informations. Au début, deux individus sont appariés, puis chaque paire de chaînes va subir un croisement. A la position i de la chaîne choisie aléatoirement, deux nouvelles chaînes seront créées après un échange de tous les caractères compris entre la position $(i+1)$et la fin de la chaîne. Ceci est illustré dans la figure 2.9, dans laquelle les chaines sont de longueur $(l = 5)$ et le

croisement est à la position ($i = 4$). Contrairement à la phase de croisement, la mutation

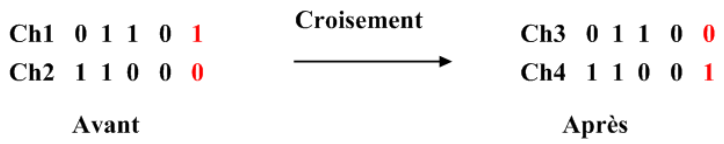

FIGURE 2.9 – Croisement en un point de deux chaînes.

est exécutée sur une seule chaîne. Elle consiste à ne modifier qu'un seul bit dans la chaîne et change un "1" à "0" et vice versa, figure 2.10, ce qui évite pour l'algorithme de stagner dans un optimum local et implique la diversité lors de la recherche des solutions [Coello.Coello et al., 2007 ; Deb, 2000].

FIGURE 2.10 – Mutation de bit dans une chaîne.

2.4.2 Paramètres de contrôle de l'algorithme

L'algorithme génétique dispose de plusieurs paramètres et éléments de contrôle qui le rendent performant et puissant. Nous pouvons citer [Koza, 1992] :

- le critère d'arrêt qui sera souvent le nombre maximal des générations,
- la taille de la population, définie par l'utilisateur et dépendant de la complexité du problème d'optimisation abordé. Avoir une idéale convergence de l'algorithme revient à bien définir ce paramètre,
- les probabilités de mutation et de croisement, variant d'un problème à un autre,
- la fonction d'adaptation des individus,
- le codage des solutions.

2.4.3 Techniques et processus d'évolution des générations

Généralement, les algorithmes génétiques sont des algorithmes générationnels, d'où la création d'une nouvelle population d'individus à chaque génération. Certains individus ne

seront pas manipulés ni pour le croisement ni pour la mutation, ce qui les rend des copies conformes aux parents.

Contrairement à la première méthode traditionnelle de reproduction, l'approche élitiste permet de ne conserver que les meilleurs individus à chaque génération et empêcher la disparition des individus de bonne qualité. Cette approche permet aussi d'accélérer la vitesse de domination imposée par les meilleurs individus [Collette & Siarry, 2004 ; Deb et al., 2002].

2.4.4 Pseudo code de l'algorithme

Plusieurs implémentations de l'algorithme génétique ont été proposées dans la littérature [Goldberg, 1994 ; Hoque et al., 2012 ; Tasan & Gen, 2012]. Le pseudo-code de l'algorithme 6 peut être retenu.

Algorithm 6 Algorithme génétique

1: Initialisation de la population
2: Evaluation de la fonction objectif
3: Calcul de l'efficacité (fonction d'adaptation)
4: **Pour** $i = 1$ à $MaxIter$ **faire**
5: Sélection aléatoire
6: Sélection proportionnelle à la valeur d'adaptation
7: Croisement
8: Mutation
9: Evaluation des fonctions objectif
10: **Fin pour**
11: **Fin Algorithme génétique**

2.4.5 Algorithmes génétiques pour l'optimisation multi-objectif

D'après K. Ded [Deb, 1999b], dans la majorité des AG en optimisation multi-objectif il s'agira de satisfaire les deux points nécessaires suivants : d'abord trouver des solutions aussi proches de la surface de compromis, c'est-à-dire converger le plus possible vers le front de Pareto, ensuite trouver un ensemble de solutions très variées, tout le long du front. Parmi les premiers algorithmes évolutionnaires d'optimisation multi-objectif, on trouve l'algorithme génétique à évaluation vectorielle VEGA, présenté en 1985 par

Schaffer [Coello.Coello et al., 2001]. Cette métaheuristique est facile à implémenter, mais son inconvénient majeur est qu'elle a tendance à générer les meilleures solutions pour un seul objectif, sans tenir compte des autres objectifs. Depuis l'apparition de l'algorithme VEGA, un nombre considérable d'algorithmes génétiques d'optimisation multi-objectif a été développé comme les algorithmes micro-GA à population réduite, les algorithmes NPGA et NSGA-II, etc. Dans ce qui suit, nous passons en revue les principes de ces algorithmes.

2.4.5.1 L'algorithme NPGA

En 1994, J. Horn et ses collègues ont proposé un algorithme évolutionnaire d'optimisation multi-objectif à sélection par tournoi basé sur la dominance de Pareto [Horn & Nafpliotis,1993 ; Horn et al., 1994]. Cet algorithme est connu sous le nom Niched-Pareto Genetic Algorithm (NPGA). Le NPGA inspire tous les concepts de l'algorithme génétique standard. La seule différence étant la méthode de sélection. Deux individus aléatoirement choisis sont comparés à un sous-ensemble de la population entière (typiquement, autour de 10 % de la population). Si l'un d'eux est dominé (par les individus choisis au hasard de la population) et l'autre ne l'est pas, alors l'individu non dominé est sélectionné pour la reproduction. Dans le cas où les deux individus sont dominés ou non dominés par l'ensemble de comparaison alors on utilise la technique de partage (*fitness sharing*) pour choisir les meilleurs individus. Dans le paragraphe suivant, nous montrons comment le calcul du compteur de niche décidera sur le résultat du tournoi.

Introduite par Goldberg et Richardson en 1987, la fonction de partage est bien étudiée plus tard par K. Deb [Goldberg & Richardson, 1987]. La fameuse difficulté pour l'algorithme génétique était liée aux individus ayant de très bonne fonction d'adaptation et donc tendance à se multiplier aux dépends des autres individus de la population. Pour les fonctions multimodales, on cherche toujours à avoir plusieurs optimums et non pas un seul. Pour cela, on introduit la fonction de partage pour bien distribuer les optimums dans l'espace de recherche. La bonne distribution des individus sera effectuée par la dégradation des fonctions d'adaptation des individus concernés, ceci est achevé via le compteur de niches [Horn & Nafpliotis, 1993 ; Kim et al., 2016]. Cette dégradation touche l'individu par rapport aux autres, dits semblables dans la population. La fonction de partage est calculée en divisant la fonction de chaque individu par le compteur de niches C_n défini par l'équation (2.5). Ce paramètre présente une estimation du nombre d'individus voisins de

l'individu concerné. Il sera aussi calculé pour tous les individus de la population courante.

$$C_n = \sum_{j \in pop} F_{sh}\left[d\left(i,\, j\right)\right] \tag{2.5}$$

où $d\left(i,\, j\right)$ est la distance entre deux individus i et j de la population, $sh\left[d\left(i,\, j\right)\right]$ est la fonction décroissante de $d\left(i,\, j\right)$ tel que : $sh\left[0\right] = 1$ et $sh\left[d \geq \sigma_{share}\right] = 0$, σ_{share} représente le rayon de niche qui est fixé toujours en fonction de la distance minimale de séparation entre les différents optimums [Salazar-Lechuga & Rowe, 2005].

Dans l'équation (2.5), $F_{sh}\left[d\right]$ est une fonction triangulaire définie l'expression suivante :

$$F_{sh}\left[d\right] = \begin{cases} 1 - \frac{d}{\sigma_{share}} & \text{si } d \leq \sigma_{share} \\ 0 & \text{sinon} \end{cases} \tag{2.6}$$

La distance entre deux individus $d\left(a,\, b\right)$ est définie comme suit :

$$d\left(a,\, b\right) = \left(\sum_{i=1}^{n} \left|a_i - b_i\right|\right)^{1/2} \tag{2.7}$$

avec x_i et y_i sont les valeurs du l'i-ème objectif,

Pour l'algorithme NPGA, la méthode de dégradation des individus n'est plus prise en compte, mais ce qui compte ici c'est le compteur de niche pour chaque individu. Comme on l'avait mentionné précédemment, on a parfois des individus qui sont ni dominés ni non dominés. Dans ce cas, l'individu qui sera choisi est celui disposant d'un petit compteur de niche. Ceci assure une sorte de diversité de solutions le long de la frontière de Pareto.

La figure 2.11, illustre la phase de sélection des individus. Il est aisé de remarquer que les individus (1 et 2) ne sont pas dominés par les autres individus. Le but étant de maintenir la diversité des solutions le long de la frontière de Pareto, l'individu ayant le plus petit compteur de niche C_n sera retenu.

D'après les études menées par C.A. Coello [Coello.Coello et al., 2001], le NPGA est un algorithme qui ne se base pas sur la technique de classement (*Pareto ranking*) de la population, ce qui rend cet algorithme le plus rapide parmi les algorithmes homologues puisqu'il est basé sur le tri d'une partie de la population [Berry & Vamplew, 2005]. Cependant, le NPGA fait appel à d'autres paramètres supplémentaires tels que la taille du tournoi et le facteur de partage. Un pseudo-code du NPGA est montré dans l'algorithme 7.

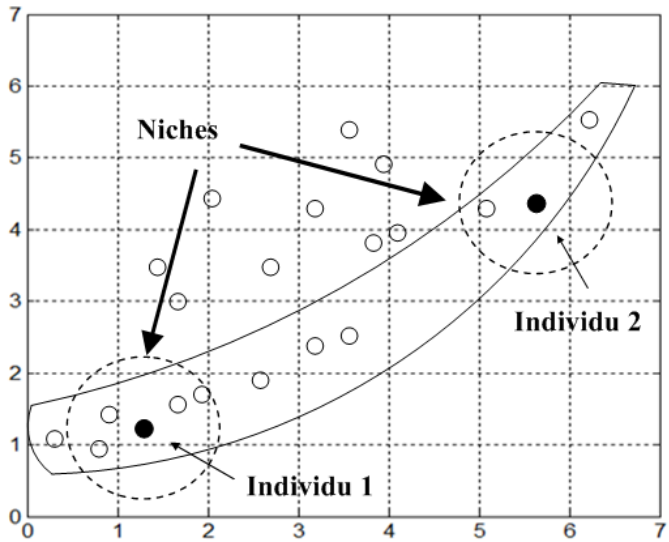

FIGURE 2.11 – Sélection d'individus avec le NPGA (d'après Horn et al. [Horn & Nafpliotis,1993]).

Algorithm 7 NPGA

1: Initialisation de la population POP
2: Evaluation des fonctions objectif
3: **Pour** $i = 1$ à $MaxIter$ **faire**
4: Sélection par tournoi entre deux individus
5: **Si** individus 1est dominé **alors**
6: Sélectionner individu 2
7: **Sinon si** individus 2 est dominé **alors**
8: Sélectionner individu 1
9: **Sinon si** individus 1 et individus 2 sont dominés ou non dominés **alors**
10: Effectuer technique de partage d'efficacité (fitness sharing)
11: Retourner l'individu qui à le petit compteur de niche (voisin)
12: **Fin si**
13: Croisement
14: Mutation
15: Evaluation des fonctions objectif
16: **Fin pour**
17: **Fin NPGA**

2.4.5.2 L'algorithme MOGA

Carlos M. Fonseca et Peter J. Fleming ont proposé une variation de la méthode de Goldberg nommée MOGA pour "Multi-Objective Genetic Algorithm" [Fonseca & Fleming, 1993 ; Van-Veldhuizen, 1999 ; Deb, 1999a]. Cette méthode est basée sur la dominance au sens de Pareto ainsi que le classement "rang" des individus. Ce dernier présente le numéro d'ordre permettant de classer un individu par rapport aux autres. En considérant un individu x_i à la génération dominé par $p_i^{(t)}$ individus, le classement ou le rang d'un tel individu est défini comme suit :

$$rang\,(x_i,\, t) = 1 + p_i^{(t)} \tag{2.8}$$

Le rang de tous les individus non dominés vaut 1, mais les autres individus non dominés disposent d'un rang plus important. Le calcul de la fonction d'adaptation pour chaque individu s'effectue sur deux étapes ; au début, il s'agit de classifier les individus selon leurs rang, ensuite, il faut affecter une valeur d'adaptation (efficacité) à chaque individu en interpolant à partir du meilleur (*rang* 1) jusqu'au plus mauvais (*rang* n).

Le pseudo-code du MOGA est présenté dans l'algorithme 8 ci-dessous :

Algorithm 8 MOGA

1: Initialisation de la population
2: Evaluation des fonctions coût
3: Assignation d'un rang basé sur les règles de dominance
4: Assignation d'une efficacité à partir du rang
5: **Pour** $i = 1$ à $MaxIter$ **faire**
6: Sélection aléatoire proportionnelle à l'efficacité
7: Croisement
8: Mutation
9: Evaluation des fonctions coût
10: Assignation d'un rang basé sur les règles de dominance
11: Assignation d'une efficacité à partir du rang
12: **Fin pour**
13: **Fin MOGA**

2.4.5.3 L'algorithme MOMGA

L'algorithme MOMGA est proposé par David A. Van Veldhuizen et Gary B. Lamont pour résoudre de problèmes multi-objectifs [Van-Veldhuizen & Lamont, 2000b]. Cet algorithme est à l'origine de l'algorithme d'optimisation monobjectif mGA.

Dans cette section, on s'intéresse à la version améliorée de l'algorithme MOMGA-II, proposée par Zydallis et al [Zydallis et al., 2001]. L'algorithme MOMGA est le seul algorithme évolutionnaire qui utilise des blocs de construction BB [Coello.Coello et al., 2007]. Contrairement aux algorithmes génétiques multi-objectifs, le MOMGA dispose d'une taille de population limitée. Celle-ci augmente qu'à la première phase d'initialisation. L'algorithme MOMGA est similaire à l'algorithme mGA qui est basé aussi sur de BBs pour trouver des solutions optimales [Van-Veldhuizen & Lamont, 2000b ; Day et al., 2004]. Le pseudo-code de l'algorithme MOMGA-II est présenté dans l'algorithme 9.

Algorithm 9 MOMGA-II

1: **Pour** $i = 1$ à k **faire**
2: Effectuer l'initialisation probabiliste complète
3: Évaluer chaque membre de la population
4: **la phase de construction des blocs**
5: **Pour** $i = 1$ à Nombre maximum de générations BBF **faire**
6: **Si** (BBF est hors l'horaire d'entrée)
7: Effectuer un filtrage des bocs de construction BBF
8: **Sinon** textEffectuer une sélection par Tournoi
9: **Fin si Fin faire**
10: **la phase de juxtaposition**
11: **Pour** $i = 1$ à Nombre maximum de générations " juxtaposition " **faire**
12: Couper et épisser
13: Évaluez chaque membre de population
14: Effectuer une sélection par Tournoi et le partage
15: $P_k(t) = P_c(t) \cup P_k(t-1)$
16: **Fin faire**
17: Mise à jour de k
18: **Fin faire**
19: **Fin MOMGA-II**

Cette implémentation est constituée de trois phases ; la phase d'initialisation, la phase

de construction des blocs et la phase de juxtaposition. Pour cette version améliorée de l'algorithme, la principale différence réside dans les deux premières phases. La phase d'initialisation consiste à utiliser l'initialisation probabiliste complète (PCI) qui permet de créer un nombre contrôlé de clones de bloc de construction d'une taille spécifiée. La deuxième phase permet de réduire le nombre de blocs de construction à travers un processus de filtrage et ensuite de stocker les meilleurs blocs de construction trouvés. Ce filtrage est réalisé avec la sélection par tournoi entre les blocs de construction. Dans la troisième phase, l'algorithme MOMGA est produit par la construction de la population grâce à l'utilisation des opérateurs "*couper*", "*épissure*" et "*recombinaison*" [Day et al., 2004 ; Goldberg al., 1989].

Dans ce pseudo-code, P_k est un archivage qui est toujours mis à jour pour conserver les meilleures solutions trouvées jusqu'à présent, P_c est l'ensemble des solutions non dominées de la génération actuelle et k est le nombre des blocs de construction.

2.4.6 Etude de l'algorithme NSGA-II

Dans cette partie, on s'intéresse plus particulièrement à l'étude de l'algorithme NSGA-II, l'une des métaheuristiques utilisée dans les présents travaux de recherche. Le choix de cet algorithme multi-objectif évolutionnaire MOEA est basé sur sa puissance et son efficacité qu'il a montrées dans plusieurs domaines et travaux [Deb et al., 2001].

2.4.6.1 Principe et origines

Cette métaheuristique d'optimisation multi-objectif est conçue par K. Deb en 2002. Elle est la version améliorée des algorithmes NSGA et NSGA-I [Deb, 1999 ; Deb et al., 2002]. Le Non-dominated Sorting Genetic Algorithm-II est considéré comme étant le meilleur et le plus efficace de ses prédécesseurs, vu qu'il ne nécessite aucun réglage de paramètres. Il utilise aussi une méthode de tri basée sur le principe de non dominance qui est plus rapide. Cet algorithme est conçu en utilisant une approche élitiste permettant de ne retenir que les meilleurs individus (solutions) au cours des générations. Avec la notion de dominance, le NSGA-II utilise une autre technique de comparaison pour calculer la distance de surpeuplement, dite aussi de "*crowding*".

L'algorithme NSGA est basé sur plusieurs niveaux de classification des individus. Avant la sélection des individus, la population est classée sur la base du non dominance. Tous les individus non dominés sont stockés dans la même catégorie. Ce groupe d'individus est

ignoré et on considère une autre couche d'individus non dominés. Ce processus se poursuit jusqu'à ce que tous les individus de la population soient classifiés. Puisque les individus dans le premier front ont les meilleures valeurs, ils obtiennent toujours plus de copies que le reste de la population. Ceci permet la meilleure recherche des régions de front optimal et mène à la convergence de la population vers ces régions. Le partage (*fitness sharing*) permet de distribuer la population sur la région, c'est-à-dire, sur le front de Pareto du problème [Srinivas & Deb, 1993].

Le NSGA a relativement bien connu du succès dans plusieurs travaux et domaines d'applications [Deb et al., 2001]. Plusieurs études comparatives ont montré la supériorité de l'algorithme NSGA-II par rapport aux algorithmes MOGA et NPGA. Le NSGA est également un algorithme très efficace en raison de la façon dont il classifie les individus. Le pseudo-code du l'NSGA est donné dans l'algorithme 10.

Algorithm 10 NSGA

1: Initialisation de la population
2: Evaluation des fonctions objectif
3: Assignation d'un rang basée sur le rang de dominance sur chaque surface de compromis
4: Calcul du compte des voisins
5: Assignation d'une efficacité partagée
6: **Pour** $i = 1$ à g **faire**
7: Sélection aléatoire proportionnelle à l'efficacité
8: Croisement
9: Mutation
10: Evaluation des fonctions objectif
11: Assignation d'un rang basée sur le rang de dominance sur chaque surface de compromis
12: Calcul du compte des voisins (compteur de niches)
13: Assignation d'une efficacité partagée
14: **Fin pour**
15: **Fin NSGA**

Le principe de l'algorithme NSGA-II est illustré dans la figure 2.12. La population $R_t = Pop_t \cup Q_t$ est la combinaison des deux populations ; la population Parents Pop_t, de taille N, et la population Enfants Q_t. Cet assemblage permet d'assurer l'élitisme de l'algorithme. La phase de tri, selon le critère de dominance choisi, aura lieu ensuite pour une taille maximale égale à $2N$. La procédure de tri permet de déterminer les n fronts de Pareto

$(F_1, F_2, ..., F_n)$. Les meilleures solutions sont appartenues aux premiers fronts. A ce stade, une nouvelle population de Parents Pop_{t+1} est formée par l'ajout des solutions trouvées au complet en ne pas dépassant la taille N. Pour cela, tant que la taille de population Parents Pop_{t+1} est inférieure à N, une procédure de "*crowding*" est appliquée sur le front F_i qui n'est pas dans Pop_{t+1}. Cette opération permet d'inclure les $|N - Pop_{t+1}|$ meilleurs individus dans la population Parents Pop_{t+1}. Comme c'est mentionné dans l'algorithme 2.5, une fois la population Pop_{t+1} est formulée, une nouvelle population Enfants Q_{t+1} est créée après les trois phases de sélection, de croisement et de mutation.

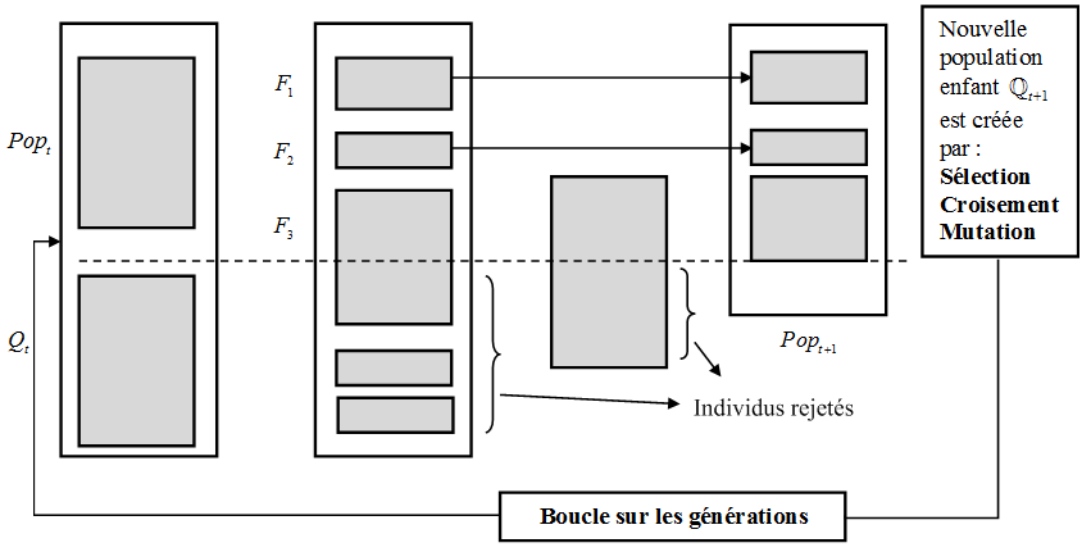

FIGURE 2.12 – Principe de l'algorithme NSGA-II [Deb et al., 2002].

2.4.6.2 Calcul de la distance de surpeuplement

Dite aussi distance de crowding, elle est calculés en fonction du périmètre formé par tous les individus voisins d'une solution comme c'est montré par la figure 2.13, qui présente le cas d'un problème bi-objectif.

Une procédure de tri des solutions dans un ordre ascendant sera lancée avant le calcul de la distance de surpeuplement. Par la suite, tous les individus ont à la fois une petite et une grande valeur de fonction objectif. Ces valeurs se voient associées à une distance infinie et pour les autres individus, pour chaque fonction objectif, on calcule la distance de surpeuplement qui est égale à la différance normalisée de toutes les valeurs de fonctions coût de deux solutions "*adjacentes*". Enfin, la distance de crowding sera la somme des distances calculées et correspondant à chaque objectif.

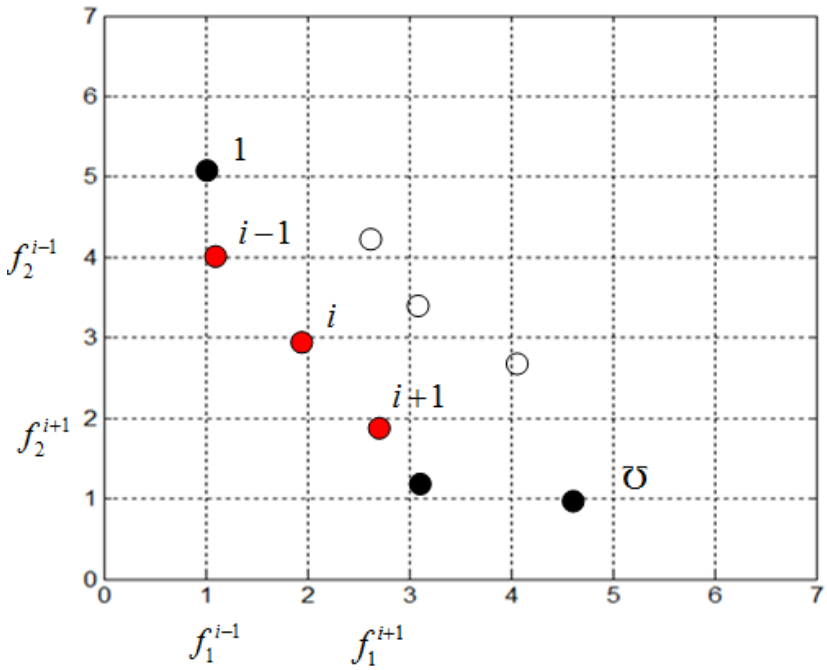

FIGURE 2.13 – Distance de crowding (surpeuplement) : cas d'un problème bi-objectif.

L'algorithme 11 montre la technique de calcul de la distance de surpeuplement de toutes les solutions non dominées S_{nd} . Les paramètres f_M^{Max} et f_M^{Min} sont respectivement les valeurs maximales et minimales de la M-ième fonction objectif et f_M^{i-1}, f_M^{i+1} sont les valeurs courantes de la même M-ième fonction objectif [Deb et al., 2002 ; Srinivas & Deb, 1993 ; Andersson & Wallace, 2002].

2.4.6.3 Pseudo-code de l'algorithme NSGA-II

Après avoir détaillé le principe théorique de l'algorithme NSGA simple, nous présentons dans l'algorithme 12 les grandes lignes de l'algorithme NSGA-II. L'opérateur $\prec_n$, appelé *crowded-comparaison*, intervient dans le processus de la sélection des individus. Défini par l'équation (2.9), cet operateur permet d'identifier la meilleure entre deux solutions. Chaque solution (i) dans l'espace de recherche est identifiée par un rang et une distance de crowding.

$$i \prec_n j \text{ si } (i_{rang} < j_{rang}) \text{ ou } ((i_{rang} = j_{rang}) \text{ et } (i_{rang} > j_{rang})) \tag{2.9}$$

Dans l'approche NSGA-II, chaque solution dispose d'un rang et celle qui possède le plus petit sera préférée. Pour deux solutions qui font partie de la même frontière de Pareto, on choisit celle qui est localisée dans la région de faible densité.

Algorithm 11 distance de crowding

1: $\mho = |S_{nd}|$ Nombre de solutions dans l'ensemble
2: **Pour** chaque i, poser $S_{nd}[i]_{dis\tan ce} = 0$ Initialiser les distances
3: **Pour** chaque objectif M
4: $S_{nd} = Trier\,(S_{nd}, M)$ Trier selon la valeur de l'objectif M
5: $S_{nd}[1]_{dis\tan ce} = \infty$
6: $S_{nd}[\mho]_{dis\tan ce} = \infty$
7: **Pour** $i = 2$ à $(\mho - 1)$**faire**
8: $S_{nd}[i]_{dis\tan ce} = S_{nd}[i]_{dis\tan ce} + \left(f_M^{i+1} - f_M^{i-1}\right)/\left(f_M^{Max} - f_M^{Min}\right)$
9: **Fin pour**
10: **Fin pour**
11: **Fin pour**
12: **Fin distance de crowding**

Algorithm 12 NSGA-II

1: **Pour** chaque itération t faire
2: $R_t = Pop_t \cup Q_t$ (Combiner les deux populations)
3: F = fast-non-dominated-sort (R_t) (Calcul de tous les fronts non dominés de R_t)
4: $Pop_t = \emptyset$, i=1
5: **Tant que** $|Pop_{t+1}| + |F_i| \leq N$ (Tant que la population n'est pas pleine) **faire**
6: $t = t + 1$
7: $Pop_{t+1} = Pop_{t+1} \cup F_i$ (Inclure le i front non dominé dans Pop_{t+1})
8: Crowding-distance-assignement $(F_i$) (Calculer la distance de "crowding" du front F_i)
9: Trier $(F_i, \prec_n)$ (Trier dans un ordre descendant en utilisant l'opérateur de comparaison $\prec_n$)
10: $Pop_{t+1} = Pop_{t+1} \cup F_i\,[1 : (N - |Pop_{t+1}|)]$ (Choisir les premiers $(N - |Pop_{t+1}|)$
11: individus du front les mieux répartis)
12: Générer une nouvelle population enfant (Q_{t+1}) par sélection, croisement et mutation
13: $t = t + 1$ (Incrémenter le compteur des générations)
14: **Fin NSGA-II**

2.5 Algorithmes d'évolution différentielle multi-objectif

2.5.1 Aperçu historique

En 1990, R. Storn et K. Price ont proposé le premier algorithme d'optimisation à évolution différentielle DE, pour Differential Evolution, afin de résoudre le problème

d'ajustement par polynômes de Tchebychev. Price et al. [Storn et al., 1997, Price & Price, 2005] ont ensuite proposé plusieurs variantes de cet algorithme. La désignation $DE/X/Y/Z$ est retenue dans ce cadre pour spécifier par "X" le mode de sélection de l'individu de référence ou l'individu cible (rand ou best) pour la mutation, par "Y" le nombre de différenciations utilisées pour la perturbation du vecteur cible et par "Z" le schéma de croisement, qui peut être exponentiel ou binomial.

Beaucoup d'études ont été faites sur les algorithmes évolutionnaires pour la résolution des problèmes d'optimisation multi-objectif. Le développement des techniques DE pour résoudre les MOP est fondé principalement dans les travaux de Nakib et al. [Nakib et al., 2010], Hammouch et al. [Hammouche et al., 2010], Bader et al. [Bader & Zitzler, 2008], Deb et al. [Deb & Tiwari, 2008] et Zitzelr et al. [Zitzler et al., 2010].

Plusieurs recherches ont montré l'extension de la DE pour résoudre les MOP dans les domaines d'optimisation continue. Abbass et al. [Abbass et al., 2001 ; Abbass, 2002] sont les premiers qui ont exploré le potentiel de la DE pour résoudre des MOP. Leur algorithme, nommé PDE pour Pareto Differential Evolution, est utilisé pour créer de nouvelles solutions. Celles non dominées sont conservées en tant qu'une base pour la génération suivante. Les résultats obtenus dans ce cadre ont montré la supériorité de l'algorithme PDE pour la résolution des MOPs de complexité accrue.

Madavan a développé dans [Madavan et al., 2002] une autre version de l'algorithme DE en utilisant un concept similaire à celui de la métaheuristique PDE d'Abbass. Cet algorithme, baptisé PDEA pour Pareto Differential Evolution Approach, applique la technique DE pour créer de nouvelles solutions et les maintenir dans une population auxiliaire.

Xue et al. [Xue et al., 2003] a introduit aussi une version de l'algorithme DE pour optimisation multi-objectif. C'est l'algorithme MODE pour Multi-Objective Differential Evolution. L'algorithme MODE utilise également la notion de Pareto et la distance de surpeuplement, mais d'une manière différente de celle de l'algorithme PDEA. La solution trouvée est ensuite utilisée pour sélectionner les meilleures solutions pour la prochaine génération.

Robic et al. [Robic & Filipic, 2005] ont proposé quelques perfectionnements de l'algorithme Multi-Objective Differential Evolution pour obtenir des bons résultats. Leur algorithme est semblable à celui de PDEA, mais avec une stratégie différente de mise à jour de la population générée. Selon cet algorithme, si la nouvelle solution générée domine la solution cible, alors il y a un remplacement immédiat de la solution cible dans la population actuelle. Cependant, si les deux sont non dominées, alors la nouvelle solution

est ajoutée à la population pour le prochain tri, sinon la solution cible est conservée. D'autres travaux sont menés dans le même cadre, nous citons les recherches d'Adeyemo et al. [Adeyemo & Otieno, 2009], d'Huang et al. [Huang et al., 2007], etc.

2.5.2 Principe et concept de base

A chaque itération, tout les individus seront mutés, ensuite croisés avec leurs mutants. Dans la première phase de mutation consiste à générer un nouvel individu v_i "*mutant individual*" pour chaque individu de la population x_i "target individus". Ceci via l'ajout la de différence pondérée d'autres individus pris aléatoirement de la population à ses composantes. Ensuite, on génère un vecteur d'essai E_i "*trial individual*" après le croisement entre l'individu et son propre mutant v_i. Enfin, la phase de sélection implique un choix entre l'individu père x_i et son descendant E_i. Celui qui est le meilleur sera conservé pour la prochaine génération. Cet enchainement sera répété à chaque itération afin de créer une nouvelle population qui à la même taille [Xiaobing et al., 2015].

2.5.2.1 Schémas de mutation

Nous présentons ci-après les schémas de mutation les plus couramment utilisés dans le formalisme d'optimisation multi-objectif par DE [Ali et al., 2012 ; Ali et al., 2009 ; Bandyopadhyay & Mukherjee, 2015]. Pour tout les schémas de mutation, $x_{best,g}$ est la meilleure solution trouvée à la génération g, les paramètres α, β, δ etϕ sont tous choisis aléatoirement de la population et ces derniers doivent être différente à chaque itération. Le facteur de poids différentiel F est compris dans l'intervalle $[0, 1]$.

- $DE/rand/1$: à la génération g et pour tout les vecteurs $x_{i,g}$, on cherche le vecteur mutant $v_{i,g}$ à partir des trois vecteurs $x_{\alpha,g}$, $x_{\beta,g}$ et $x_{\chi,g}$ choisis aléatoirement de la population initiale. Un paramètre F est défini pour contrôler l'amplitude du vecteur d'exploration $(x_{\beta,g} + x_{\chi,g})$. Ce schéma est défini par l'expression suivante :

$$v_{i,g} = x_{\alpha,g} + F\left(x_{\beta,g} - x_{\chi,g}\right) \tag{2.10}$$

- $DE/rand/2$: on choisi aléatoirement de la population cinq autres vecteurs différents de $x_{i,g}$. Le vecteur $v_{i,g}$ est créé par l'équation suivante :

$$v_{i,g} = x_{\alpha,g} + F\left(x_{\beta,g} - x_{\chi,g}\right) + F\left(x_{\delta,g} - x_{\phi,g}\right) \tag{2.11}$$

- $DE/best/1$: on ajoute au meilleur individu trouvé une perturbation via deux individus choisis aléatoirement de la population. Donc le nouvel individu sera créé par

l'équation (2.12) :

$$v_{ig} = x_{best,g} + F\left(x_{\beta,g} - x_{\chi,g}\right) \tag{2.12}$$

– $DE/best/2$ le vecteur mutant $v_{i,g}$ est produit par l'ajout d'une perturbation au meilleur individu via cinq autres individus choisis aléatoirement de la population. Ce schéma de mutation est exprimé par l'équation suivante :

$$v_{i,g} = x_{best,g} + F\left(x_{\beta,g} - x_{\chi,g}\right) + F\left(x_{\delta,g} - x_{\phi,g}\right) \tag{2.13}$$

– $DE/current\ to\ best/1$ exprimé par l'équation (2.14), le vecteur mutant $v_{i,g}$ est créé à l'aide de deux autres vecteurs choisis au hasard ainsi que le meilleur vecteur de la génération courante :

$$v_{i,g} = x_{i,g} + F\left(x_{best,g} - x_{i,g}\right) + F\left(x_{\alpha,g} - x_{\beta,g}\right) \tag{2.14}$$

Après la phase de la mutation, un croisement binaire aura lieu pour constituer le vecteur d'essai final $e_{i,g}$ selon le vecteur $x_{i,g}$ de la population à la génération g et le vecteur mutant correspondant $v_{i,g}$:

$$e_{i,j} = \begin{cases} v_{i,j} \text{ si } rand(0,1) \leq TC \text{ ou } j = j_{rand} \\ x_{i,j} \text{ sinon} \end{cases} \tag{2.15}$$

Le taux de croisement TC permet d'avoir la distance séparant le vecteur d'essai $u_{i,g}$ du vecteur de référence $x_{i,g}$. Pour un faible taux de croisement, la majorité des composantes de $e_{i,g}$ sont identiques à celles de référence. Mais, avec un taux de croisement proche de 1, le vecteur d'essai $e_{i,g}$ sera identique au vecteur mutant $v_{i,g}$. Ce dernier peut être situé loin du vecteur de référence, selon le schéma de mutation, d'où une bonne d'exploration de l'espace de recherche [Adeyemo & Otieno, 2009 ; Zhaoa et al., 2016].

2.5.2.2 Opérateur de sélection

Lorsque que la phase de mutation est terminée, le croisement, défini par l'équation (2.15), sera effectué et suivi par la phase de sélection. Cette étape est la plus importante pour les problèmes multiobjectifs, vu que la sélection rigoureuse des solutions candidates aide à la production d'un front de Pareto optimal.

L'algorithme MODE combine les concepts de l'algorithme PDEA [Madavan et al., 2002] et DEMO [Robic & Filipic, 2005], mais avec une légère différence. Afin d'expliquer le processus de sélection de MODE, on donne d'abord une courte description du processus de sélection pour les deux algorithmes PDEA et DEMO.

L'algorithme PDEA applique le DE pour créer de nouvelles solutions de taille NP et les maintenir. Ensuite, il combine les deux populations, d'où la taille totale de l'ensemble après la combinaison égale $2NP$. Après cela, des solutions de tailles NP seront sélectionnées sur la base du classement par la non dominance et la distance de surpeuplement pour la prochaine génération. Cet algorithme permet une vérification globale de la non dominance des solutions parent et enfant "*descendant*" à la fois, même si elle nécessite un effort de calcul supplémentaire dans le tri des combinés.

Pour l'algorithme DEMO [Robic & Filipic, 2005], la solution d'essai remplace la solution cible si elle la domine. Si la solution cible domine la solution d'essai, la solution d'essai est rejetée. Dans le cas contraire, la solution d'essai est ajoutée à la population. Ainsi, à la fin d'une génération, la taille totale de la population se situe entre NP et $2NP$. Cette population est tronquée pour l'étape suivante de l'algorithme. Le processus de troncature est basé sur le tri selon la non dominance. La procédure de troncature ne conserve que les meilleures solutions (*élites*) de taille NP dans la population.

Dans l'algorithme MODE, la solution d'essai est comparée à la solution cible, si elle domine la solution cible, elle remplace la solution cible immédiatement dans la population courante (comme dans DEMO) et la solution cible est ajoutée à la population "avancée", sinon une nouvelle solution d'essai est ajoutée à cette dernière. Après chaque génération, les deux populations (*courante et avancée*) sont combinés, dont la taille totale est égale $2NP$ (comme dans PDEA). En outre, l'algorithme MODE intègre également une stratégie de conservation, empruntée de l'algorithme NSGA-II [Deb et al., 2002] pour la troncature des solutions $2NP$ à NP.

Le remplacement immédiat de la solution mère par le candidat qui la domine est le noyau de l'algorithme MODE. La solution nouvellement créée qui pénètre dans la population prend instantanément part à la création de nouvelles solutions. Cela contribue à la réalisation du premier objectif de l'optimisation multi-objectifs , qui est la convergence vers un bon front de Pareto. Au cas où, il y a beaucoup de fronts dans la population courante et la solution cible appartient à la première frontière et une solution d'essai la domine, alors elle est rejetée. Cependant, au lieu de rejeter , elle est stockée dans la population "*avancée*" pour le dernier tri, car elle peut se placer dans le front ultérieur de la population pour la prochaine génération.

2.5.2.3 Pseudo-code de l'algorithme MODE

Le pseudo-code présenté dans l'algorithme 13 décrit la méthode MODE proposée.

Algorithm 13 MODE

1: Initialiser n, k, $npop$, f, TC, $NbObj = 0$ et entrer $x_{\min}[n]$ et $x_{\max}[n]$
2: Générer les $Npop$ solutions aléatoirement
3: **Pour** $\{i, k\}$ de 0 à $Npop$ **faire**
4: $x[i][k] = x_{\min}[k] + (x_{\max}[k] - x_{\min}[k]) * u[0,1]$ **Fin pour**
5: Générer les $Npop$ solutions opposées
6: **Pour** $\{i, k\}$ de 0 à $Npop$ **faire**
7: $y[i][k] = x_{\min}[k] + x_{\max}[k] - x[i][k]$ **Fin pour**
8: Evaluer les valeurs de la fonction pour $2 * Npop$ solutions
9: **Pour** i de 0 à $2 * Npop$ **faire**
10: **Objective function()** $NbObj = NbOBj + 1$
11: Sélectionnez n solutions en utilisant la dominance et la distance de surpeuplement et les stocker dans $Npop_1$
12: pop_1 =nondominated CrowdSort(X,Y)
13: **Tant que** $(NbObj < \max\_fun)$ **faire**
14: **Pour** i de 0 à $Npop$ **faire**
15: Sélectionner aléatoirement trois individus $x_{\alpha,g}$, $x_{\beta,g}$ et $x_{\chi,g}$
16: $v_i = x_{best,g} + F(x_{\beta,g} - x_{\chi,g})$
17: Sélectionner le meilleur vecteur non dominé pour la phase de mutation
18: Générez l'individu en utilisant l'équation de mutation
19: Générer le vecteur d'essai final par l'opération de croisement entre x_i et v_i avec l'équation (2.12)
20: Evaluer la valeur de fonction à v_i
21: **Objective Function ()** $NbObj = NbObj + 1$
22: Vérifier de la non dominance entre le vecteur d'essai final u_i et x_i
23: **Si** (u_i domine x_i) **alors**
24: Remplacer x_i par e_i dans la population actuelle $Npop_1$
25: Ajouter x_i dans la population actuelle $Npop_2$
26: **Sinon** Ajouter e_i dans la population actuelle $Npop_2$ **Fin pour**
27: Sélectionner les npop solutions les plus convenables en utilisant la non dominance et le tri selon la distance de surpeuplement, et les stocker dans $Npop_1$
28: $Npop_1$ **=non domination Crowd sort(**$Npop_1$ **,** $Npop_2$**)**
29: **Fin tant que**
30: **Fin MODE**

2.6 Métriques de performance en optimisation multi-objectif

Les performances des méthodes d'optimisation multi-objectif sont mesurées via des métriques. Ces métriques permettent aussi de visualiser les caractéristiques des ensembles de solutions obtenues [Van-Veldhuizen & Lamont, 2000a]. Dans ce qui suit, nous passons en revue les plus utilisées de ces métriques.

2.6.1 Distance générationnelle

Cette métrique permet de mesurer la distance entre les solutions trouvées et celles qui font partie de la surface de compromis. La distance générationnelle est définie comme suit :

$$GD \triangleq \frac{1}{n_e}\left(\sum_{i=1}^{n_D}(d_i)^p\right)^{1/p} \tag{2.16}$$

où p est un paramètre constant égal à 2 et d_i présente la distance entre une telle solution i et la solution la plus proche "*voisine*" appartenant à la surface de compromis optimale TPF. La surface de compromis optimale sera notée TPF, pour Theoretical Pareto Frontier ou "*surface de compromis théorique*" et l'ensemble de solutions obtenu à l'itération i sera noté $PF_{current}(i)$, pour Pareto Front Current ou "*surface de compromis courante*", n_e étant le nombre d'éléments dans l'ensemble de solutions, noté $PPF(i)$, pour Practical Pareto Frontier ou "*surface de compromis pratique*" qui est l'union des solutions non dominantes de chaque ensemble $PF_{current}(i)$. Le PPF ne contient pas des solutions dominées [Van-Veldhuizen, 1999] :

$$PPF(i) = PF_{current}(i) \cup PPF(i-1) \tag{2.17}$$

$$PPF(1) = PF_{current}(1) \tag{2.18}$$

2.6.2 Espacement

La métrique espacement, notée , décrit numériquement la répartition des vecteurs dans PPF et mesure ainsi la distance de variation des vecteurs voisins de ce PPF [Coello.Coello et al., 2007 ; Van-Veldhuizen, 1999]. L'expression de cette métrique est exprimée comme

suit :

$$S \triangleq \sqrt{\frac{1}{n_e - 1} \sum_{i=1}^{n} \left(\bar{d} - d_i\right)^2} \tag{2.19}$$

avec :

$$d_i = \min_{j} \sum_{k=1}^{K} \left| f_k^i - f_k^j \right|, \ i = 1, \ 2, \ ..., \ n \tag{2.20}$$

$\bar{d}$ est la moyenne de toutes les distances d_i.

2.6.3 Rapport d'erreur

La métrique Rapport d'Erreur (ER) calcule le pourcentage des solutions non dominées trouvées et qui ne sont pas dans le front Pareto optimal (surface de compromis) [Collette & Siarry, 2004 ; Van-Veldhuizen , 1999]. L'ensemble des solutions converge vers la surface de compromis quand la valeur de cette métrique est très proche de 1. Cette métrique est définie dans l'équation (2.21) :

$$ER \triangleq \frac{1}{n} \sum_{i=1}^{n} e_i \tag{2.21}$$

avec $e_i = 0$ si la solution appartient à la surface de compromis et $e_i = 1$ dans le cas contraire.

2.6.4 Conclusion

Ce chapitre a présenté les définitions de base et les notations formelles des algorithmes évolutionnaires en optimisation multi-objectif pour une future exploitation dans la synthèse et l'optimisation de structures de commande robuste RST de systèmes complexes. Les définitions d'un problème d'optimisation multi-objectif sous contraintes, de la dominance des solutions ainsi que des concepts d'optimalité au sens de Pareto sont détaillées. Ce chapitre a présenté aussi une analyse approfondie des MOEAs, les exemples des algorithmes génétiques NSGA, et puis NSGA-II, ainsi que les algorithmes à évolution différentielle MODE. Les métriques de mesure de performance des algorithmes métaheuristiques en optimisation multi-objectif, en l'occurrence la distance générationnelle, l'espacement, le rapport d'erreurs et la diversité, sont également décrites pour être utilisées dans la suite de nos travaux dans la comparaison des algorithmes proposés.

Le chapitre suivant fera l'objet de l'approche MOPSO proposée en optimisation multi-objectif à base des essaims particulaires. Les perfectionnements effectués sur un tel algo-

rithme d'optimisation d'intelligence en essaim seront présentés et mis en exergue. La mise en oeuvre pour l'optimisation d'un benchmark de fonctions de test issues de la littérature sera d'abord envisagée. Une étude comparative avec les algorithmes NSGA-II et MODE déjà étudiés sera évoquée.

Chapitre 3

Perfectionnement d'un algorithme d'optimisation par essaim particulaire multi-objectif

3.1 Introduction

La recherche d'une méthode systématique pour l'ajustement des paramètres d'un régulateur polynomial à structure canonique RST est le principal objectif de ces travaux de recherche. Comme on a mentionné dans les chapitres précédents, le problème de synthèse d'une telle structure de commande numérique peut devenir délicat avec la complexité des systèmes dynamiques à commander d'un côté, et le calcul lourd ainsi que le temps de résolution prohibitif des techniques classiques proposées d'autre côté. Face à ces difficultés, nous proposons la formulation et la résolution de ce type de problème de synthèse robuste par une approche d'optimisation par essaim particulaire dans un formalisme multi-objectif. Cette approche est choisie vu ses utilisations fréquentes dans divers domaines de l'ingénierie et pour plusieurs problèmes relevant de l'automatique et de l'informatique [Gangulya et al., 2013 ; Shokrian & Ann Highb, 2014 ; Bouallègue et al., 2012]. Le perfectionnement d'une telle approche, lui a fait une solution prometteuse pour la résolution du problème de synthèse des régulateur RST.

Dans ce chapitre, on détaille l'algorithme d'optimisation par essaim particulaire, alias PSO pour Particle Swarm Optimization, qui constitue l'outil principal de résolution des problèmes de synthèse et de réglage RST formulés dans cette thèse. Au début, l'algorithme PSO sera présenté avec toutes ces caractéristiques. Ensuite, on décrit l'ap-

proche développée avec tous les outils utilisés dans la conception de l'algorithme d'optimisation multi-objectif MOPSO. Puis, les perfectionnements réalisés ainsi que d'autres améliorations relevant de ce cadre sont particulièrement soulignés. Enfin, une étude comparative avec les algorithmes évolutionnaires, vus dans le chapitre précédent, sera effectuée et discutée.

3.2 L'optimisation par essaim particulaire

Dans cette section, on présente la technique d'optimisation par essaim particulaire monobjectif dans le but d'extraire tous ces principes pour pouvoir l'adapter dans un formalisme multi-objectif et l'utiliser pour la synthèse de lois de commande robuste à structure numérique RST.

3.2.1 Formulation mathématique

On considère un espace de recherche de dimension D. Une particule i de l'essaim est modélisée par un vecteur de position $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ et un vecteur de vitesse noté $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$. Chaque particule garde en mémoire la meilleure position par laquelle elle est passé, notée $Pbest_i = (pbest_{i1}, pbest_{i2}, ..., pbest_{iD})$. Toutes les meilleures positions atteintes par l'essaim sont notées par $Gbest_i = (gbest_{i1}, gbest_{i2}, ..., gbest_{iD})$. Au début du processus de recherche des solutions, toutes les particules de l'essaim sont initialisées aléatoirement dans l'espace de recherche. Ensuite, à chaque itération toute particule de l'essaim se déplace, en combinant linéairement les trois composantes de l'équation (3.1). A l'itération t, le vecteur de vitesse est calculé comme suit [Kennedy & Eberhart, 1995a ; Eberhart et al., 1996] :

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 [pbest_{ij}(t) - x_{ij}(t)] + c_2 r_2 [gbest_{ij}(t) - x_{ij}(t)] \tag{3.1}$$

avec $j \in \{1, 2, ..., D\}$ et w une constante, appelée coefficient d'inertie.

Le degré d'attraction vers la meilleure position d'une particule et celle de ces informatrices est représenté respectivement par deux coefficients de confiance cognitif c_1 et social c_2. La bonne exploration des particules dans l'espace de recherche est garantie par les coefficients r_1 et r_2 qui sont deux nombres aléatoires tirés uniformément dans l'intervalle $[0, 1]$.

L'équation (3.1) est composée de trois termes qui sont : $wv_{ij}(t)$ une composante physique de déplacement dont w est un paramètre variable permettant de bien contrôler le

déplacement de la particule à la prochaine itération, $c_1 r_1 \left[pbest_{ij}(t) - x_{ij}(t)\right]$ une composante cognitive de déplacement et $c_2 r_2 \left[gbest_j(t) - x_{ij}(t)\right]$ une composante sociale de déplacement [Dréo et al., 2003]. La position à l'itération $t+1$ de la particule i est alors définie par l'équation suivante :

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t)\, j \in \{1, 2, ..., D\} \tag{3.2}$$

Comme déjà mentionné, l'algorithme PSO est à base de population. En effet, toutes les particules sont initialisées d'une façon aléatoire dans l'espace de recherche de dimension D. A chaque itération, les particules se déplacent suivant les équations (3.1) et (3.2) dites de mouvement, figure 3.1. Une fois que le changement des positions a eu lieu, une mise à jour affecte les deux vecteurs $Pbest_i$ et $Gbest_i$. A l'itération $t+1$, ces deux derniers seront mis à jour selon les deux équations (3.3) et (3.4).

$$Pbest_i(t+1) = \begin{cases} Pbest_i(t), & \text{si } f(x_i(t)) \geq f(Pbest_i(t)) \\ x_i(t), & \text{sinon} \end{cases} \tag{3.3}$$

$$Gbest_i(t+1) = \text{Arg} \min_i \left[Pbest_i(t+1)\right] \tag{3.4}$$

Dans la littérature, le critère d'arrêt d'un algorithme PSO peut être définit de manières différentes. L'algorithme est exécuté tant que l'un des trois, ou tous à la fois, des critères de convergences suivants sont vérifiés [Clerc & Siarry, 2009] :

- le nombre maximum d'itération défini n'est pas atteint ;
- la variation de la vitesse des particules est proche de zéro ;
- la valeur de l'objectif est satisfaisante vis-à-vis à la relation suivante :

$$\left|f\left(gbest_{ij}(t)\right) - f\left(gbest_{ij}(t-\alpha)\right)\right| \leq \varepsilon, \text{ avec } \alpha \in [1, \Lambda] \tag{3.5}$$

Le paramètre ε représente une tolérance choisie le plus souvent de l'ordre de 10^{-5} et Λ un nombre d'itérations choisi de l'ordre de 10.

3.2.2 L'algorithme PSO

Dans cette section, on présente dans l'ordre les différentes étapes d'un algorithme PSO dans le cas de l'optimisation mono-objectif. La version ainsi décrite sera adoptée dans toutes les futures formulations de problèmes d'optimisation envisagées. Aussi, elle sera à la base de l'algorithme MOPSO développé dans un formalisme d'optimisation multi-objectif :

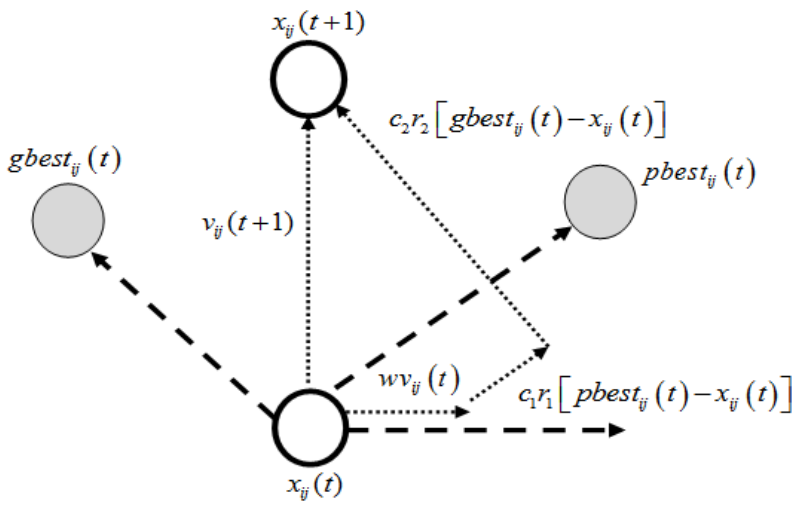

FIGURE 3.1 – Illustration du déplacement d'une particule dans l'espace de recherche en combinant les trois tendances du mouvement.

i. Au début, tous les paramètres de l'algorithme sont déclarés ; les coefficients cognitif c_1 et social c_2, le facteur d'inertie w, la taille de la population qui est normalement définie selon le problème d'optimisation posé.

ii. Chaque particule dispose d'une position x_0 et une vitesse v_0 qui sont initialisées aléatoirement pour être mises à jour conformément aux équations (3.1) et (3.2).

iii. Evaluer toutes les particules de la population dans l'espace de recherche pour pouvoir déterminer les vecteurs des meilleures positions initiales $Pbest_0$ et $Gbest_0$.

iv. Calculer les positions et les vitesses des particules à chaque itération conformément aux équations de mouvement (3.1) et (3.2). Puis, évaluer les positions des particules et mettre à jour les grandeurs $Pbest$ et $Gbest$.

v. Répéter la quatrième étape tant que le critère d'arrêt n'est pas atteint.

Ces étapes peuvent être résumées par le pseudo code de la technique PSO illustré dans l'algorithme 14.

3.2.3 Confinement des particules

Le long de la recherche des solutions, chaque particule risque de se déplacer rapidement, cela conduirait à sortir du domaine de recherche initialement défini. D'après [Eberhart et al., 1996], on peut introduire un autre paramètre, noté $V_{\max}$, pour limiter la vitesse des toutes les particules dans l'essaim. Le contrôle de la vitesse évite ainsi la divergence de

Algorithm 14 PSO

1: Initialiser aléatoirement les positions et les vitesses des N particules de l'essaim
2: Evaluer les positions des particules dans l'espace de recherche
3: **Pour** 1 à N **faire**
4: $Pbest_i = x_i$
5: Calculer le vecteur $Gbest$ selon (3.4)
6: **Fin pour**
7: **Tant que** le critère d'arrêt n'est pas attient **faire**
8: Déplacer les particules selon les équations (3.1) et (3.2)
9: Evaluer les positions des particules
10: Mettre à jour $Pbest_i$ et $Gbest_i$ selon les équations (3.3) et (3.4)
11: **Fin tant que**
12: **Fin PSO**

l'algorithme et assure un compromis entre la diversification et l'intensification d'une telle métaheuristique. La définition de l'intervalle $[\![V_{\min}, V_{\max}]\!]$ sur les vitesses de déplacement peut limiter la distance parcourue par une telle particule à chaque itération. D'après [Clerc & Siarry, 2009], on peut définir des contraintes sur les variables de décision du problème traité comme suit :

$$x_{j\,\min} \leq x_{ij} \leq x_{j\,\max} \tag{3.6}$$

Dans le cas où une particule sort de cet intervalle, un mécanisme de confinement est mis en oeuvre pour attribuer à cette particule la valeur de la position de frontière la plus proche. Ceci revient à changer l'équation de mouvement (3.2) par l'équation (3.7) suivante :

$$x_{ij} = \mathrm{Min}\left(\mathrm{Max}\left(x_{ij} + v_{ij},\, x_{\min}\right), x_{\max}\right) \tag{3.7}$$

D'après [Clerc et al., 2009], l'équation de la vitesse est remplacée dans ce cas par l'expression suivante :

$$v_{ij} = \gamma v_{ij}, \text{avec } \gamma \in [\![-1,\, 0]\!] \tag{3.8}$$

Dans d'autres études, les particules qui se trouvaient, lors du processus de recherche, à l'extérieur seront laissées en dehors de l'espace de recherche sans évaluer les fonctions objectif associées. Néanmoins, la particule peut être bloquée à la frontière initialement défini et annuler tous les paramètres associés à sa vitesse.

3.2.4 Paramètres de l'algorithme PSO

On définit dans cette section tous les paramètres qui interviennent dans l'algorithme PSO. Le choix de ces paramètres est toujours l'étape la plus délicate et la plus importante pour l'optimisation. Ce choix a une influence sur la robustesse et les performances en convergence de l'algorithme vers les meilleures régions de l'espace de recherche.

3.2.4.1 Coefficients de confiance

Les coefficients de confiance cognitif c_1 et social c_2, dits aussi constantes d'accélération, pondèrent respectivement les tendances de la particule à vouloir suivre sa propre meilleure position ou celle de ses informatrices. Dans la littérature, une diversité d'approches, le plus souvent empiriques, a été proposée pour le choix des coefficients de confiance. Une technique de choix et de réglage de ces paramètres, proposée par Clerc et Siarry [Clerc & Siarry, 2009], consiste à choisir ces coefficients en relation avec le facteur d'inertie w, de préférence choisi de l'ordre de 0.72, selon la formule suivante :

$$c_1 = c_2 = \frac{(w+1)^2}{2} \tag{3.9}$$

ou bien indépendamment du facteur d'inertie, pour avoir une convergence parfois un peu plus lente, mais plus sûre, selon la formule suivante :

$$c_1 = c_2 = 1.19 \tag{3.10}$$

Les analyses mathématiques menées par F. Van Den Bergh [Van Den Bergh, 2002], dans le cadre de la recherche de conditions théoriques pouvant conclure sur la convergence de l'algorithme PSO, ont abouti à la relation (3.11) qui donne, en effet, une idée sur le choix de tels paramètres.

$$0 \leq c_1 + c_2 \leq 4 \tag{3.11}$$

L'étude du réglage des différents paramètres de l'algorithme PSO sort du cadre de notre travail. Nous adoptons des valeurs vérifiant la relation (3.11) ci-dessus. Typiquement, nous choisissons , conformément aux travaux menés dans [Shi & Eberhart, 1998a ; Shi & Eberhart, 1999 ; Kennedy et al., 2001].

3.2.4.2 Coefficient de constriction

Les recherches pour améliorer l'algorithme PSO conduisent à introduire ce paramètre dont le but est d'éviter la divergence de l'algorithme [Kennedy, 1998 ; Ozcan & Mohan,

1999 ; Van Den Bergh, 2002]. D'après [Clerc & Kennedy, 2002 ; Kennedy, 2001], la convergence de l'algorithme PSO est toujours liée aux trois paramètres c_1, c_2 et w. En outre, la combinaison entre ces dernières variables peut donner un certain équilibre pour les phases d'intensification et de diversification de l'algorithme. Pour cela, un nouveau paramètre, dit coefficient de constriction, est mis en jeu afin de mieux garantir la convergence de l'algorithme. L'équation (3.1) est réécrite alors comme suit :

$$v_{ij}(t+1) = \chi\left[v_{ij}(t) + \phi_1 r_1 \left(pbest_{ij}(t) - x_{ij}(t)\right)\right] + \phi_2 r_2 \left[gbest_j(t) - x_{ij}(t)\right] \tag{3.12}$$

avec :

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}},\ \phi = (\phi_1 + \phi)_2 > 4 \tag{3.13}$$

et, par analogie avec la version (3.1) de l'algorithme PSO, nous pouvons déduire que :

$$\chi = w,\ c_1 = \chi\phi_1\, et\, c_2 = \chi\phi_2 \tag{3.14}$$

Les valeurs adoptées dans la majorité des travaux et proposées par [Clerc & Kennedy, 2002] sont $\phi = 4.1$ et $\phi_1 = \phi_2$, ce qui donne un coefficient de constriction égal à $\chi = 0.7298$. Cette méthode évite la convergence prématurée et mène à la bonne exploration des différentes régions de l'espace de recherche. Dans ce cas, il est recommandé de fixer le paramètre $V_{\max}$ comme suit [Eberhart et al., 2001] :

$$V_{\max} = \frac{(x_{\max} - x_{\min})}{2} \tag{3.15}$$

Cette méthode n'est pas la seule pour garantir la bonne convergence de l'algorithme PSO et l'exploration de l'espace de recherche. D'autres études sont proposées et validées dans les travaux de [Van Den Bergh, 2001 ; Peer et al., 2003 ; Trelea, 2003].

3.2.4.3 Facteur d'inertie

Ce paramètre est introduit par Y. Shi et R. Eberhart dans la première version de l'algorithme PSO [Shi & Eberhart, 1998]. Le but était de bien contrôler la vitesse trouvée à chaque itération. Le choix judicieux du facteur d'inertie provoque une bonne exploration de l'espace de recherche. Ainsi, ce facteur apparait important, vu qu'il contrôle et évite l'intensification et l'exploration locales de l'algorithme. En effet, une valeur adéquate du facteur d'inertie permet de gérer l'amplitude de vitesse d'une particule donnée. Lors du développement de l'algorithme MOPSO, on s'est basé sur les études empiriques menées par Y. Shi et R. Eberhart dans [Shi & Eberhart, 1998a ; Shi & Eberhart, 1998b ; Shi et al.,

1999 ; Shi et al., 1997] pour choisir la valeur adéquate du paramètre . Dans notre travail, on adopte un facteur d'inertie comme une constante positive $w \in [\![0.4, 1.2]\!]$.

Le facteur d'inertie a une grande influence sur la découverte des nouvelles régions de l'espace de recherche, pour cela Y. Shi et R. Eberhart [Shi & Eberhart, 1998] ont pensé, comme une seconde amélioration, à un facteur d'inertie de valeur variable et qui décroit au cours du temps dans l'intervalle $[\![0.4, 0.9]\!]$. Au début du processus de recherche, on accorde une valeur élevée à w_t pour une bonne exploration de l'espace de recherche. Ensuite, on diminue cette dernière pour concentrer la recherche dans les meilleures régions déjà explorées. La technique la plus répandue et la plus simple, adoptée dans nos travaux, consiste à considérer une décroissance linéaire du facteur d'inertie au cours des itérations, comme introduite dans [Chan & Tiwari, 2007 ; Shi et al., 1999] :

$$w_{t+1} = w_{\max} - \left(\frac{w_{\max} - w_{\min}}{t_{\max}}\right) t \tag{3.16}$$

où $t_{\max}$ représente le nombre maximum des itérations de l'algorithme,$w_{\min} = 0.4$ et $w_{\max} = 0.9$ désignent respectivement les valeurs minimale et maximale du facteur d'inertie, choisies conformément à [Shi & Eberhart, 1998]. Dans [Chatterjee & Siarry, 2006], les auteurs ont utilisé une stratégie d'évolution non-linéaire pour définir un facteur d'inertie dynamique. Dans [Eberhart & Shi, 2001b], une autre variante, dans laquelle le facteur d'inertie est choisi au hasard selon une distribution uniforme dans l'intervalle $[\![0.5, 1]\!]$, a été proposée. Le choix d'un tel intervalle a été inspiré de la version de l'algorithme PSO avec facteur de constriction, proposée par M. Clerc et J. Kennedy.

3.2.5 Topologie du voisinage

La métaheuristique PSO, comme technique d'optimisation stochastique, est inspirée d'un comportement social d'où l'importance de la notion de voisinage. Dans la littérature du PSO, plusieurs topologies de voisinage ont été définies pour bien spécifier envisager les relations dans l'essaim. Dans ce sens, on distingue deux techniques de voisinage : social et géographique.

Le voisinage social est le plus utilisé en pratique car il est simple à mettre en oeuvre, peu coûteux en temps de calcul et tend à devenir un voisinage géographique en cas de convergence de l'algorithme PSO. Le voisinage géographique, dit aussi physique, est basé sur le calcul de la distance euclidienne pour chaque particule lors du processus de recherche et à chaque itération. Dans la littérature, nous distinguons essentiellement les topologies en étoile, en anneau et en cercle, conformément à la figure 3.2 :

– Topologie en cercle : dite aussi en anneau et dans laquelle chaque particule communique avec n_v voisines immédiates ($n_v = 2$). On parle de la version locale de l'algorithme PSO.
– Topologie en rayon : les particules de l'essaim ne communiquent qu'avec une seule particule dite centrale. Seule cette dernière se déplace vers la meilleure position. L'échange d'informations avec le voisinage n'aura lieu que si ce déplacement apporte une amélioration.
– Topologie en étoile : chaque particule est reliée à toutes les autres particules de l'essaim qui y sont considérées comme informatrices. Tout l'essaim représente un voisinage. On parle de la version globale de l'algorithme PSO. Par ailleurs, il convient de noter que la topologie en étoile permet une convergence plus rapide de l'algorithme PSO que celle en anneau en plus d'une simplicité remarquable de mise en oeuvre. Par contre, cette dernière est moins vulnérable aux optima locaux. Généralement, les performances de l'une ou de l'autre de ces topologies dépendent du problème d'optimisation traité [Poli et al., 2007].

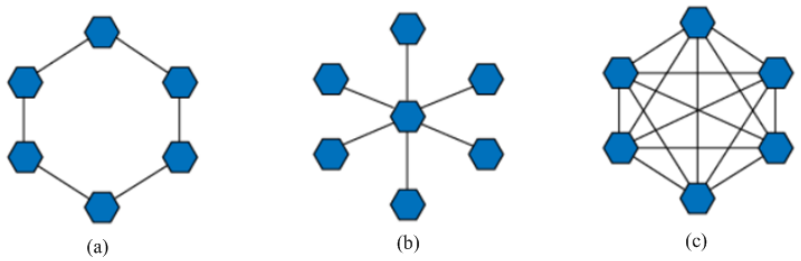

FIGURE 3.2 – Différents types de topologie pour un essaim de particules : (a) en cercle, (b) en rayon, (c) en étoile.

Pour encore améliorer les performances de la méthode, d'autres topologies, dites adaptatives ou dynamiques, peuvent être adoptées [Collette & Siarry, 2004].

3.2.6 Hybridations de l'algorithme PSO

On peut définir l'hybridation d'un algorithme d'optimisation comme la combinaison des caractéristiques de deux techniques où on pourrait extraire les bénéfiques des deux à la fois. Généralement, les stratégies d'hybridation sont utilisées pour des objectifs de perfectionnement des approches d'optimisation élaborées. L'hybridation est connue dans plusieurs métaheuristiques dont l'algorithme PSO est l'un des plus touché. L'algorithme PSO a

connu la première hybridation en 1998 dans les travaux d'Angeline avec des concepts inspirés des algorithmes évolutionnaires [Angeline, 1998]. Une hybridation avec les algorithmes génétiques a été introduite dans les travaux de Robinson [Robinson et al., 2002]. Cette combinaison est justifiée par l'efficacité montrée par l'algorithme génétique dans la phase d'intensification. Miranda & Fonseca ont proposé dans [Miranda & Fonseca, 2002] une hybridation entre les stratégies évolutionnaires et l'algorithme PSO. Dans [Shelokar et al., 2004], les auteurs on proposé la combinaison de l'algorithme PSO avec celui de colonies de fourmis, cette hybridation a mené à l'amélioration remarquable du processus d'intensification de la métaheuristique hybride ainsi obtenue. Les auteurs dans [Zhang et al., 2009] ont proposé une hybridation entre les métaheuristiques PSO et DE, baptisée la technique DE-PSO, qui est devenue une approche très utilisée dans beaucoup de travaux. Cette nouvelle technique est basée sur les stratégies de mutation fournies par l'approche DE dont le but était d'améliorer la convergence de l'algorithme résultant.

3.3 Algorithme MOPSO proposé

3.3.1 Description de l'approche

Dans cette section, on détaille l'algorithme d'optimisation MOPSO proposé dans un formalisme d'optimisation multi-objectif. Comme on a déjà mentionné, on s'est inspiré des concepts de base de l'algorithme PSO monobjectif pour le développement de la technique MOPSO. Afin d'appliquer la stratégie PSO pour résoudre des problèmes d'optimisation multi-objectif, plus particulièrement le problème de synthèse de la commande polynomiale RST. La solution d'un problème d'optimisation multi-objectif ne consiste pas en une solution unique, comme pour le cas des problèmes monocritères, mais plutôt un ensemble des solutions [Coello.Coello & Lechuga, 2002 ; Collette & Siarry, 2004].

En effet, l'optimisation multi-objectif mène à l'obtention d'un ensemble des solutions différentes dites ensemble de Pareto optimal, ou optimal au sens de Pareto. En général, lors de la résolution d'un problème d'optimisation multicritères à base de la technique PSO, les principaux objectifs à atteindre sont les suivants [Reyes-Sierra & Coello.Coello, 2006 ; Coello.Coello et al., 2004] :

- maximiser le nombre d'éléments de l'ensemble optimum de Pareto trouvé ;
- minimiser la distance entre le front Pareto produit par l'algorithme MOPSO par rapport au front de Pareto optimal préalablement connu ;

– maximiser la distribution des solutions trouvées.

Compte tenu que l'algorithme PSO est basé sur le concept de population, il est souhaitable de produire plusieurs solutions non dominées à chaque itération. Donc, comme avec tout autre algorithme évolutionnaire, les trois principaux points, à prendre en considération lors de l'extension de l'algorithme PSO à l'optimisation multi-objectif, sont les suivants :

– comment sélectionner les particules, à utiliser en tant que dirigeants, afin de donner la préférence à des solutions non dominées par rapport à ceux qui sont dominées ?
– comment conserver les solutions non dominées, trouvées au cours du processus de recherche, en tenant compte de toutes les solutions non dominées par rapport à toutes les solutions à l'itération précédente ? En outre, il est souhaitable que ces solutions soient bien réparties le long du front Pareto ;
– comment maintenir la diversité dans l'essaim afin d'éviter la convergence vers une solution unique ?

Comme nous l'avons mentionné dans la section précédente, la résolution des problèmes d'optimisation monobjectif, le leader que chaque particule utilise pour mettre à jour sa position est déterminé une fois une topologie de voisinage est établie. Cependant, dans le cas des problèmes d'optimisation multi-objectif, chaque particule peut avoir un ensemble de différents leaders dont un seul peut être sélectionné dans le but de mettre à jour sa position. Cet ensemble des leaders est généralement stocké dans un endroit différent de l'essaim, que nous appellerons archive externe. Toutes les solutions non dominées trouvées sont stockées dans cette archive. Les solutions contenues dans l'archive externe sont utilisées en tant que dirigeants lorsque les positions des particules de l'essaim doivent être mises à jour. En outre, le contenu de l'archive externe sera la sortie finale de l'algorithme MOPSO.

Plus précisément, nous décrivons ci-dessous toutes les étapes de l'algorithme MOPSO proposé pour la résolution de problèmes d'optimisation multi-objectif formulés plus loin dans nos travaux [Martinez & Coello Coello, 2011 ; Carvalho & Pozo, 2012 ; Moslemi & Zandieh, 2011] :

1. initialiser la population des particules,
   (a) **Pour** i de 1 à Pop **faire**
   (b) Initialiser x_i
2. initialiser la vitesse de chaque particule v_i,
   (a) **Pour** i de 1 à Pop **faire**

(b) Initialiser $v_i = 0$

3. évaluer chaque particule dans la population Pop,
4. mémoriser les postions des particules qui présentent des vecteurs non dominés dans l'archive A,
5. générer les hyper-cubes de l'espace de recherche exploré. Il s'agit de localiser les particules en utilisant des hyper-cubes comme un système de coordonnés dans lequel les coordonnés de chaque particule sont définis selon les valeurs de ses fonctions objectifs,
6. initialiser la mémoire de chaque particule. Cette mémoire, qui sert à guider les particules pour se déplacer à travers l'espace de recherche, est stockée dans l'archive :
   Pour i de 1 à Pop **faire**
   $Pbest_i = x_i$
7. Tant que le nombre maximum d'itérations n'est pas atteint Faire

   (a) Calculer la vitesse de chaque particule :

$$v_i = wv_i + c_1 r_1 \left(Pbest_i - x_i\right) + c_2 r_2 \left(A_h - x_i\right) \tag{3.17}$$

   avec A_h est une valeur prise à partir de l'archive.
   L'indice h est choisi de la façon suivante : les hyper-cubes contenant plus d'une particule se voient attribuer une fitness égale au résultat de la division d'un nombre $N > 1$ par le nombre des particules qu'ils contiennent. Ensuite, on applique la sélection par roulette en utilisant les valeurs de la fitness pour choisir l'hyper-cube dont nous prendrons la particule correspondante. Une fois l'hyper-cube est sélectionné, on choisit au hasard une particule. [Chiu et al., 2007 ; Reyes-Sierra & Coello.Coello, 2006].

   (b) Calculer les nouvelles positions des particules en ajoutant les vitesses produites trouvées aux positions de l'étape précédente :

$$x_i = x_i + v_i \tag{3.18}$$

   (c) Maintenir les particules dans l'espace de recherche dans le cas où elles vont au-delà de leurs frontières et éviter de générer des solutions qui ne se trouvent pas dans l'espace de recherche valide. Quand une variable de décision va au-delà de frontières de l'espace de recherche, alors soit on prend la valeur de la limite

correspondante (limite inférieure ou limite supérieure) ou bien, la vitesse de la particule sera multipliée par -1 de sorte qu'elle effectuera des recherches dans le sens opposé.

(d) Evaluer chacune des particules dans la population.

(e) Mettre à jour le contenu de l'archive avec la représentation géographique des particules dans les hyper-cubes [Herrero et al., 2008 ; Reyes-Sierra & Coello.Coello, 2006]. Cette mise à jour consiste à insérer toutes les positions actuellement non-dominées dans l'archive. Toutes les positions, solutions dominées du référentiel, sont éliminées dans le processus de recherche. Comme la taille de l'archive est limitée, chaque fois qu'il est plein on applique un critère secondaire de rétention. Les particules situées dans les zones les moins peuplées de l'espace de recherche sont prioritaires par rapport à ceux situées dans les régions très peuplées.

(f) Lorsque la position actuelle de la particule est meilleure que la position mémorisée, elle sera mise à jour comme suit :

$$Pbest_i = x_i \tag{3.19}$$

Le critère utilisé pour décider quelle position, déjà mémorisée, devrait être retenue consiste tout simplement à appliquer la dominance de Pareto [Shokrian & Ann Highb, 2014]. Si la position actuelle est dominée par celle mémorisée, alors la position dans la mémoire est conservée. Sinon, cette position actuelle remplace celle en mémoire. Si aucune des positions n'est dominée, alors on sélectionne au hasard une position parmi l'ensemble mémorisé.

(g) Incrémenter le compteur de boucle et refaire les étapes précédentes.

8. **Fin tant que**

3.3.2 Archive externe

Plusieurs méthodes d'optimisation multi-objectif utilisent cette notion d'archive parmi lesquelles on peut citer les algorithmes MOPSO, PAES, SPEA, etc. Le traitement et la mise à jour des solutions non dominées trouvées à chaque itération nécessite un endroit de stockage et de sauvegarde appelé archive externe. Ce dernier est capable d'accueillir toutes les nouvelles solutions non dominées et de rejeter à la fois d'autres solutions devenues dominées. Malgré l'efficacité de cette approche, plusieurs auteurs se mettent d'accord sur

le fait qu'elle est coûteuse en terme de complexité et de mise en oeuvre. En effet, la taille de l'archive peut devenir importante. Ce paramètre est toujours proportionnel à la complexité du problème abordé [Reyes-Sierra & Coello.Coello, 2006 ; Coello.Coello et al., 2007].

3.3.3 Contrôleur d'archive

Le rôle du contrôleur d'archive est de décider si une nouvelle solution devrait être ajoutée à l'archive ou non. L'objectif principal de cet espace de stockage externe est de garder un historique des vecteurs non dominés trouvés le long du processus de recherche.

Le processus de décision consiste à comparer les vecteurs non dominés trouvés à chaque itération conformément au principes illustrés dans la figure 3.3 [Martinez & Coello.Coello, 2011 ; Coello.Coello & Lechuga, 2002] :

- si l'archive est vide, alors la solution actuelle est acceptée, Figure 3.3-cas 1 ;
- si cette nouvelle solution est dominée par une autre solution de l'archive externe, alors une elle sera automatiquement rejetée, Figure 3.3-cas 2, sinon aucun des éléments contenus dans la population externe ne domine la solution qui souhaite entrer ;
- s'il y a des solutions dans l'archive qui sont dominées par de nouveaux éléments, alors ces solutions sont retirées de l'archive, Figure 3.3-cas 3 et 4 ;
- enfin, si la population extérieure a atteint sa capacité maximale autorisée, alors la procédure de grille adaptative, détaillée ci-après, est introduite, Figure 3.3-cas 5.

3.3.4 Grille adaptative

Pour produire des fonts de Pareto qui ne sont formés que par des solutions non dominées bien réparties, l'algorithme MOPSO proposé utilise une grille adaptative. L'idée de base est d'utiliser une archive externe pour stocker toutes les solutions non dominées. Dans ce casr l'espace des fonctions objectif est divisé en région, figure 3.4. La particule de la population externe peut être insérée en dehors des limites actuelles de la grille. Cette grille sera recalculée de nouveau à chaque ajout de solution potentielle, Figure 3.5.

Ainsi, la grille adaptative est un espace formé par des hyper-cubes pouvant être interprétés comme des régions géographiques ayant plusieurs solutions [Reyes-Sierra & Coello.Coello, 2006 ; Coello.Coello & Lechuga, 2002 ; Coello.Coello et al., 2004].

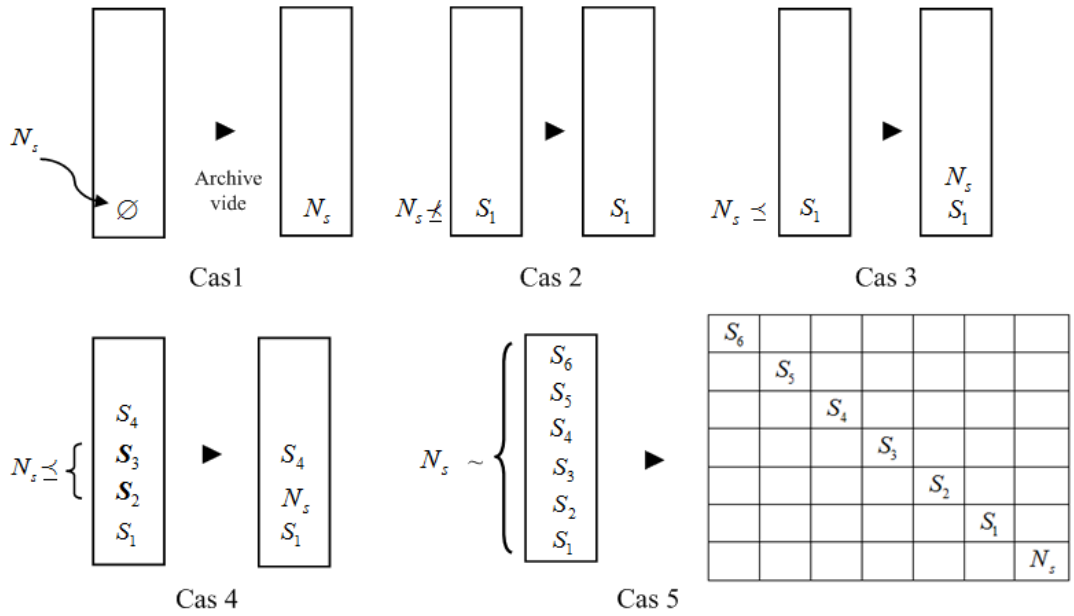

FIGURE 3.3 – Les cas possibles pour le contrôleur d'archive.

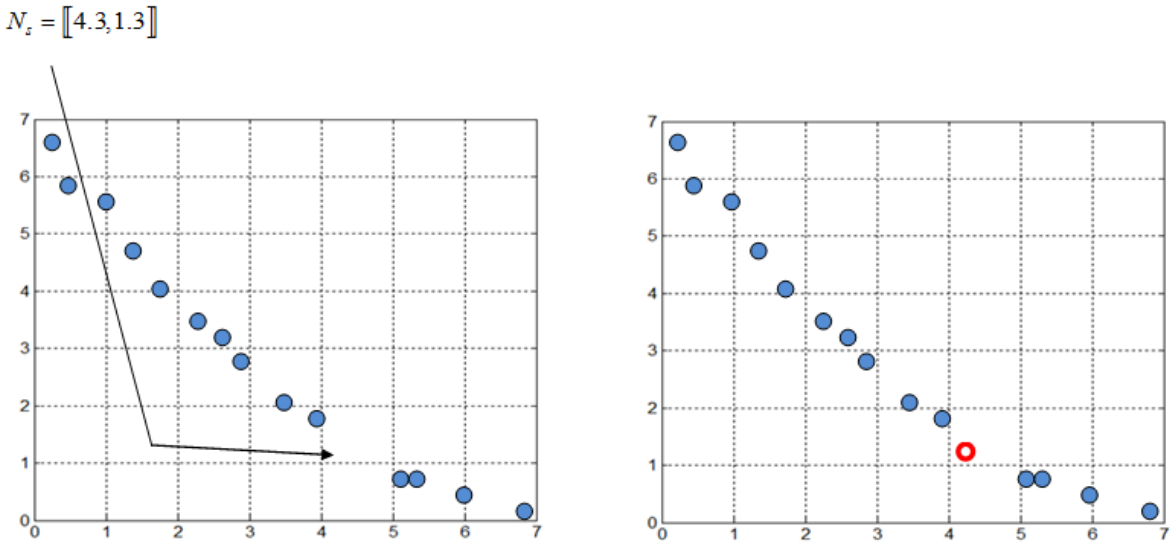

FIGURE 3.4 – Représentation graphique de l'insertion d'une nouvelle solution dans la grille adaptative lorsque la particule se situe dans les frontières de la grille.

3.3.5 Pseudo-code de l'algorithme

Dans cette partie, on donne le pseudo-code de la méthode d'optimisation MOPSO présentée par l'algorithme 3.2. À la suite de la mise à jour des positions des différentes particules dans l'algorithme MOPSO proposé, il est possible que ces dernières se trouvent

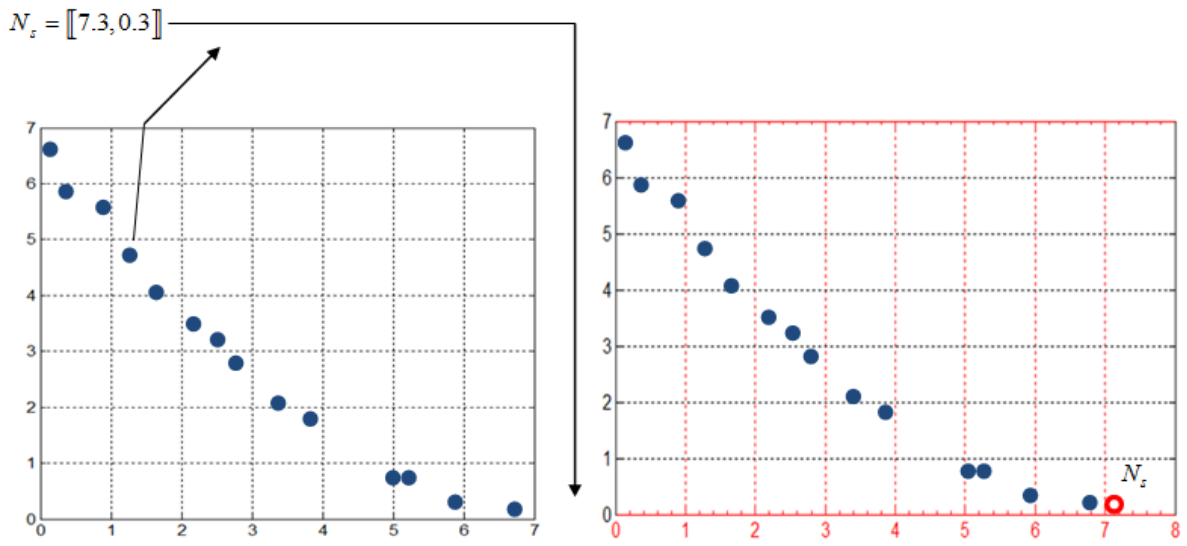

FIGURE 3.5 – Représentation graphique de l'insertion d'une nouvelle solution dans la grille adaptative lorsque la particule se trouve en dehors des frontières de la grille.

Algorithm 15 MOPSO

1: $A = \emptyset$ archive initialement vide
2: $\{x_i, v_i, Gbest_i, Pbest_i\}_{i=1}^{npop} = Init\,()$ initialiser aléatoirement les positions et les vitesses
3: **Tant que** $t < t_{\max}$ **faire** $t_{\max}$nombre maximum d'itérations
4: **Pour** i de 1 à $npop$ **faire**
5: **Pour** j de 1 à D **faire** mise à jour des vitesses et des positions
6: $v_{ij} = wv_{ij} + c_1 r_1 \left(Pbest_{ij} - x_{ij}\right) + c_2 r_2 \left(Gbest_{ij} - x_{ij}\right)$
7: $x_{ij} = x_{ij} + v_{ij}$ **Fin Pour**
8: $x_i = \text{imposer\_contraintes}\,(x_i)$
9: $y_i = f(x_i)$ évaluer les objectifs
10: **Si** $x_i \underline{\nprec} N_s \,\forall\, N_s \in A$ **alors** ajouter les solutions x_i non dominées dans A
11: $A = \left\{N_s \in A |\, N_s \underline{\nprec} x_i\right\}$ retirer les solutions dominées par les x_i
12: $A = A \cup x_i$ ajouter les solutions x_i dans A
13: **Fin si Fin Pour**
14: **Si** $x_i \underline{\nprec} Pbest_i \,\forall \left(x_i \underline{\nprec} Pbest_i \wedge Pbest_i \underline{\nprec} x_i\right)$ **alors** mise à jour de $Pbest_i$
15: $Pbest_i = x_i$ **Fin Si**
16: $Gbest_i = selectionner\_Guide\,(x_i, A)$
17: **Fin Tant que**

en dehors de la région des solutions réalisables. Dans ce cas, on doit les contraindre pour revenir à la région réalisable moyennant la fonction *imposercontraintes*() de la ligne 9 de l'algorithme 15.

3.4 Perfectionnement de l'algorithme MOPSO proposé

Dans ce paragraphe, on propose l'utilisation d'une technique, nommée epsilon-dominance, pour améliorer les performances de l'algorithme MOPSO en optimisation multi-objectif. Cette approche d'amélioration permettra de bien choisir les meilleures solutions optimales durant le processus de recherche de l'algorithme MOPSO. Le recours à ce type de perfectionnement étant justifié par la diversité des solutions obtenues sur le front optimal de Pareto ainsi que le temps de résolution, moins prohibitif, du problème multi-objectif traité.

3.4.1 Méthode d'epsilon-dominance

Dans le formalisme d'optimisation multi-objectif, une archive des solutions admissibles, pour construire l'ensemble Pareto, doit être maintenue. Afin de déterminer si une particule doit être inclue dans l'archive, la méthode la plus fréquemment adoptée consiste à conserver toutes les solutions non dominées conformément à la dominance de Pareto. L'inconvénient d'une telle approche est le contrôle de la taille de l'archive qui peut rapidement devenir très grande et difficile à maintenir. D'autres chercheurs ont proposé l'utilisation d'autres formes de dominance [Laumanns et al., 2002a ; Mostaghim & Teich, 2003]. La principale et la plus adoptée pour les algorithmes MOPSO étant l'epsilon-dominance [Deb et al., 2005]. Le principe de base de cette technique est présenté par les définitions suivantes :

Définition 3.1 : Le vecteur de décision $x_1 \in D$ est dit ε-domine le vecteur de décision $x_2 \in D$, noté $x_1 \prec_{\in} x_2$, si pour $\varepsilon > 0$ on a :

$$\frac{f_i(x_1)}{(1+\varepsilon)} \leq f_i(x_2) \qquad \forall\, i = 1, \ldots, M \tag{3.20}$$

Définition 3.2 :(Front ε-Pareto approximatif) :
Soit $F \subseteq \Re^m$ l'ensemble des vecteurs solutions et $\varepsilon > 0$. Le front de Pareto $F_\varepsilon \subseteq F$ est

appelé un ε-Pareto approximatif de F s'il contient tous les vecteurs $x_1 \in F$ qui sont non ε-dominés par aucun vecteur $x_2 \in F_\varepsilon$:

$$\forall x_2 \in F : \exists x_1 \in F_\varepsilon \text{ telque} x_1 \prec_\varepsilon x_2 \tag{3.21}$$

La taille de l'epsilon-archive à l'instant t est limitée par le terme suivant :

$$Nb = \left(\frac{\log B}{\log(1+\varepsilon)} \right)^{m-1} \tag{3.22}$$

avec B est la valeur maximale de l'objectif.

L'utilisation principale du concept d'epsilon-dominance est de filtrer les solutions non dominées dans l'archive externe, comme montré dans la figure 3.6. A gauche de cette figure, on peut observer la zone dominée par une solution quelconque du front de Pareto et à droite, la zone dominée se voit prolongée par une valeur proportionnelle au paramètre ε défini par l'utilisateur [Laumanns et al., 2002b ; Deb et al., 2005 ; Coello.Coello et al., 2007]. En utilisant le concept de l'epsilon-dominance, on définit un ensemble de boîtes

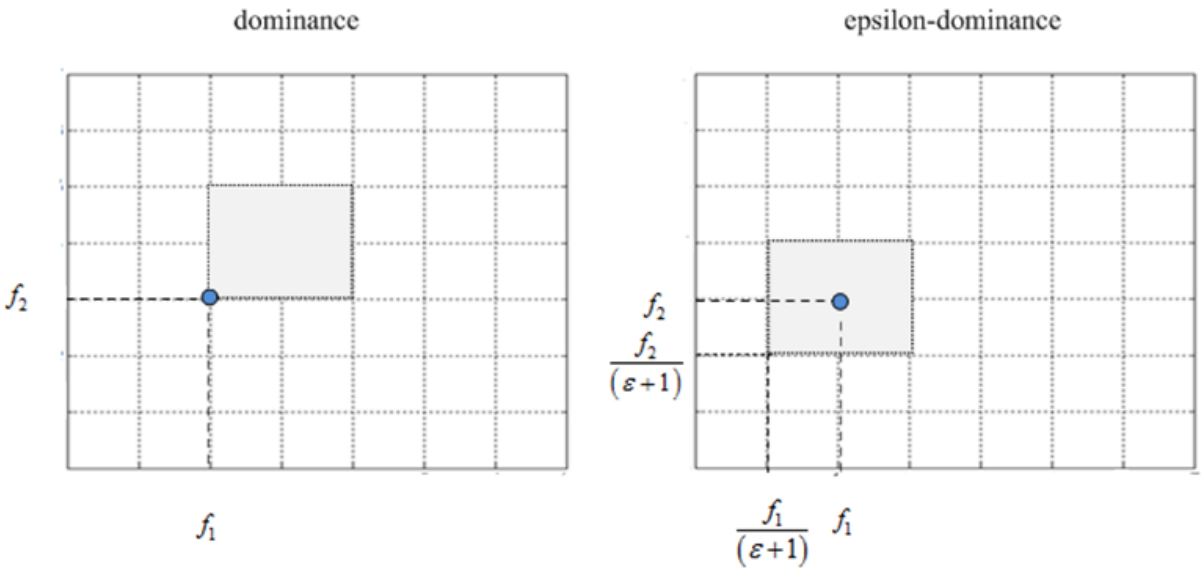

FIGURE 3.6 – Concept de la technique d'epsilon-dominance en optimisation multi-objectif.

de taille ε et une seule solution non dominée sera retenue. Pour toutes ces boîtes, les solutions qui sont plus proches du coin inférieur gauche seront choisies. Ceci est bien illustré dans la figure 3.7 pour un cas d'optimisation bi-objectif. L'utilisation de l'epsilon-dominance garantit que les solutions retenues sont non dominées par rapport à toutes les solutions générées. Il est intéressant de noter, cependant, que lors de l'utilisation de

l'epsilon-dominance, la taille de l'archive finale externe dépend de la valeur d'epsilon, qui est normalement un paramètre défini par l'utilisateur.

La figure 3.7 est un exemple de l'utilisation de technique epsilon-dominance dans une archive externe. La solution 1 domine la solution 2, elle sera alors préférée au détriment de la solution 2. Les solutions 3 et 4 sont incomparables. Cependant, la solution 3 est meilleure que la solution 4, puisqu'elle est plus proche de l'angle inférieur gauche représenté par le point $(2\varepsilon, 2\varepsilon)$. La solution 5 domine la solution 6, donc la solution 5 est préférée. La solution 7 n'est pas alors acceptée. Cette solution, représentée par le point $(2\varepsilon, 3\varepsilon)$, est dominée par la case définie par le point $(2\varepsilon, 2\varepsilon)$.

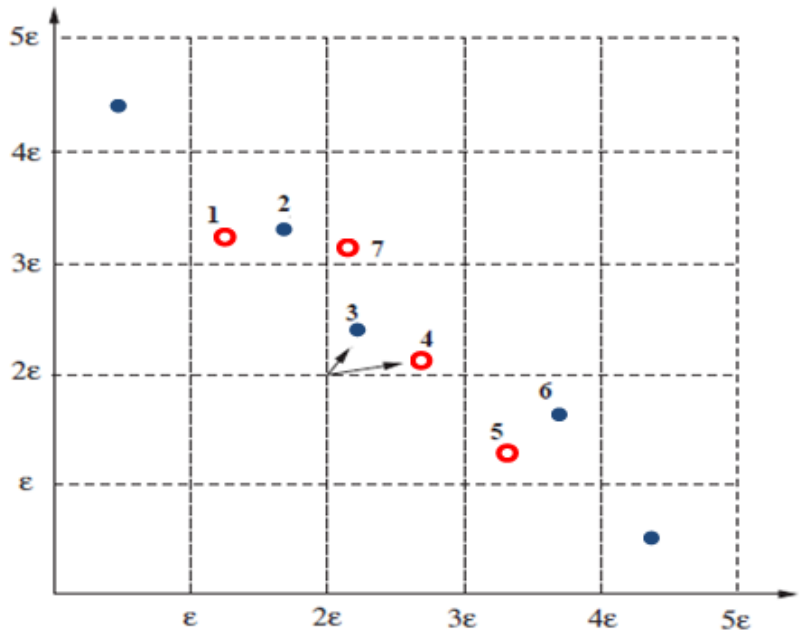

FIGURE 3.7 – Illustration du choix des solutions non dominées dans l'epsilon-Archive.

3.4.2 Algorithme epsilon-MOPSO proposé

Après avoir défini les concepts théoriques de l'approche ε-dominance en vue de son exploitation pour le perfectionnement de l'algorithme MOPSO initialement introduit, nous présentons dans l'algorithme 16 un pseudo-code de la technique ε-MOPSO proposée :

Algorithm 16 Epsilon-MOPSO

1: initialiser l'essaim (définir la position et la vitesse de chaque particule)
2: initialiser les guides, envoyer les guides dans l'epsilon-archive noté $\varepsilon - A$
3: $iter = 0$
4: **Tant que** $iter \leq iter_{\max}$ **faire**
5: calculer la grille
6: **Pour** chaque particule **faire**
7: sélectionner les guides de la grille
8: mise à jour des positions des particules
9: muter les positions des particules
10: évaluer les objectifs
11: mise à jour de $Pbest$
12: **Fin Pour**
13: mise à jour des guides, envoyer les guides dans l'epsilon-archive $(\varepsilon - A)$
14: **Si** $(\varepsilon - A_{size} > \varepsilon - A_{\max size})$ **alors**
15: retirer des particules de l'epsilon-archive $(\varepsilon - A)$
16: **Fin Si**
17: $iter = iter + 1$
18: **Fin Tant que**
19: **Epsilon-MOPSO**

3.5 Mise en oeuvre et comparaison avec les techniques NSGA-II et MODE

Cette section porte essentiellement sur la présentation et l'interprétation des résultats de simulation obtenus pour les techniques d'optimisation multi-objectif MODE, NSGA-II, MOPSO et ε-MOPSO ainsi proposées. Ces algorithmes sont testés sur un benchmark de fonctions de test issues de la littérature sur l'optimisation multi-objectif et définies en Annexe.

Les figures 3.8, 3.9, 3.10 et 3.11 montrent les résultats obtenus avec ces algorithmes pour les fonctions de test F1, F2, F3 et F4 choisies. Les paramètres des algorithmes sont présentés en Annexe. Les vrais fronts de Pareto des problèmes traités son également représentés. Les tableaux 3.2, 3.3 et 3.4 présentent les résultats relatifs aux métriques de performance DG, SP et ER décrites précédemment. Pour les deux premiers tests, on

peut constater que la performance moyenne de l'algorithme MOPSO est la meilleure en ce qui concerne les deux métriques, distance générationnelle DG et taux d'erreur ER, par rapport aux algorithmes MODE et NSGA-II. Mais pour la première fonction, la métrique espacement SP pour l'algorithme MOPSO est légèrement élevé par comparaison avec l'algorithme NSGA-II.

Pour le deuxième test, l'espacement pour l'algorithme MOPSO est légèrement inférieur que l'algorithme NSGA-II. Mais, le front de Pareto produit par l'algorithme MODE montre que les solutions non dominées trouvées ne sont pas en mesure de couvrir l'ensemble du front de Pareto optimal. D'un autre côté, on peut noter que l'algorithme MOPSO est plus rapide en convergence vers le meilleur front de Pareto que les autres pour toutes les fonctions test retenues, Tableau 3.1.

TABLE 3.1 – Temps de simulation en seconde des algorithmes proposés.

Algorithmes d'optimisation	F1	F2	F3	F4
MOPSO	74.451	37.801	70.771	63.412
NSGA-II	161.124	147.147	122.184	114.356
MODE	102,077	89,144	113,090	107,911

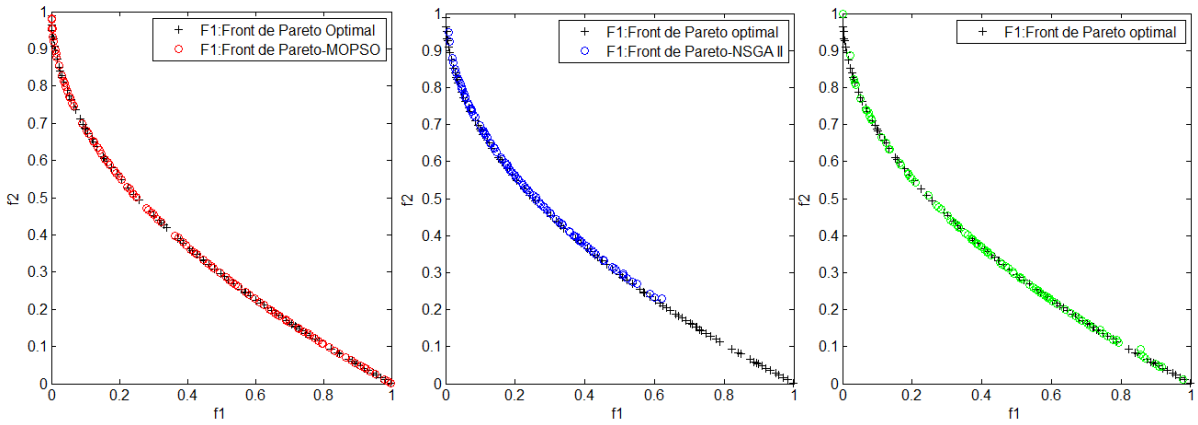

FIGURE 3.8 – Fronts de Pareto produits pour la fonction de test F1.

D'après les tableaux ci-dessous, on peut constater que la performance moyenne de l'algorithme MOPSO pour la fonction de test F3 est la meilleure pour la métrique distance de génération DG. Elle est légèrement inférieure pour le taux d'erreur ER produit par l'algorithme MODE. L'algorithme NSGA-II a le meilleur espacement, mais ces solutions

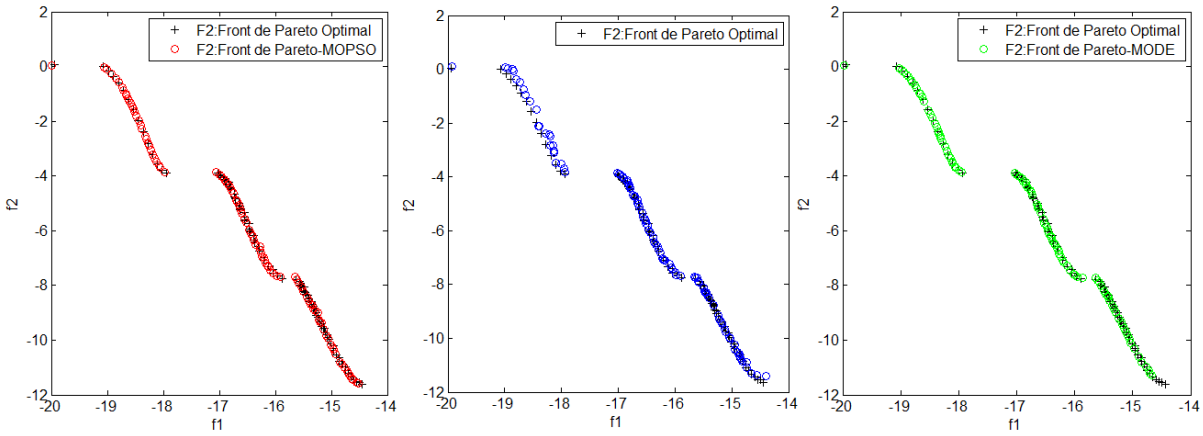

FIGURE 3.9 – Fronts de Pareto produits pour la fonction de test F2.

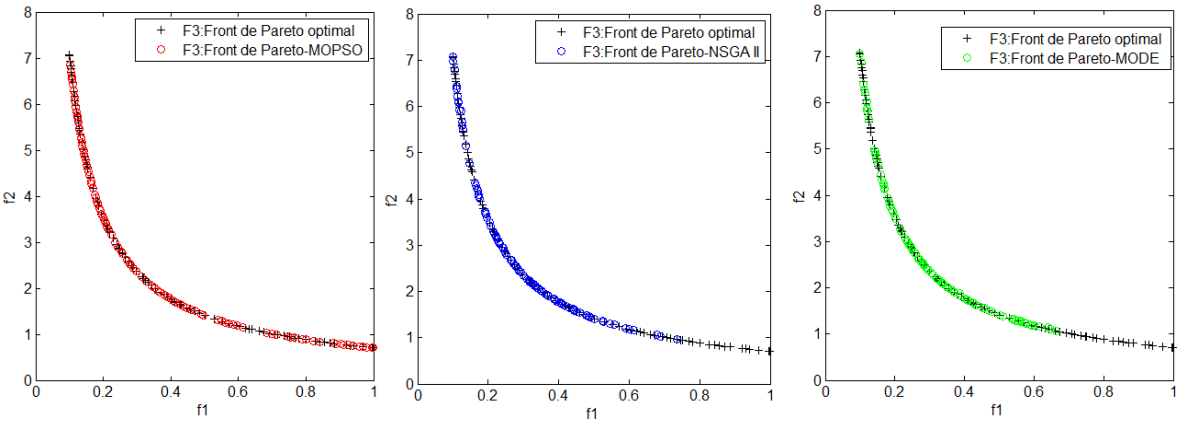

FIGURE 3.10 – Fronts de Pareto produits pour la fonction de test F3.

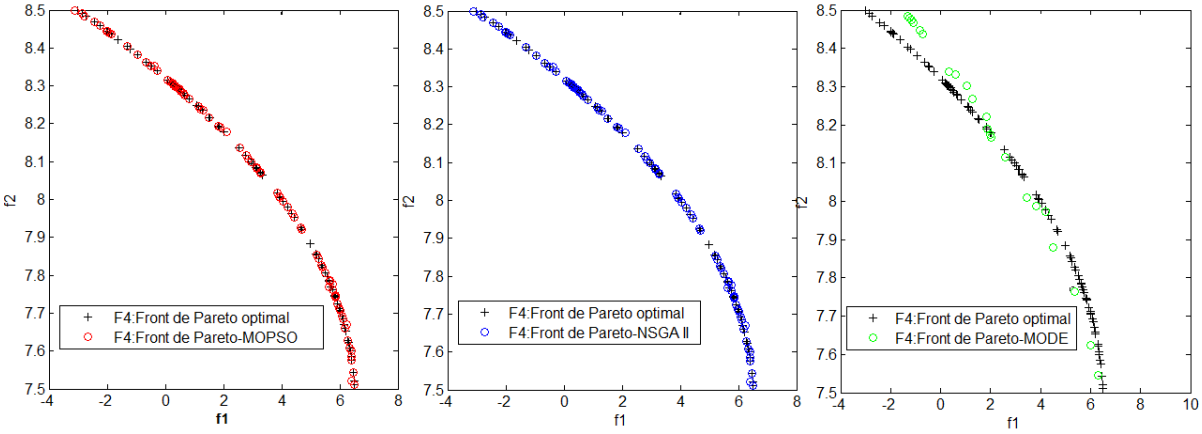

FIGURE 3.11 – Fronts de Pareto produits pour la fonction de test F4.

non dominées ne couvrent pas toute la frontière de Pareto. Pour le quatrième test F4, on peut constater que la performance moyenne de l'algorithme MOPSO est la meilleure pour le taux d'erreur et la distance de génération. En ce qui concerne l'espacement, cet algorithme est légèrement supérieur par rapport à l'algorithme MODE et avec un écart type plus inférieur. En regardant les fronts de Pareto de cette fonction de test, il est facile de remarquer que pour les algorithmes MOPSO et NSGA-II les solutions obtenues ont permis de couvrir tout le front de Pareto. Ceci est un exemple dont lequel une métrique peut être trompeuse, car la métrique d'espacement fournit une bonne valeur pour l'algorithme MODE et les solutions non dominées produites par l'algorithme ne font pas partie du vrai front de Pareto du problème. En outre, il est important de noter que le temps de simulation de l'algorithme MOPSO est à peu près la moitié du temps pris par les autres algorithmes, Tableau 3.1.

Table 3.2 – La métrique GD obtenu pour les fonctions de test.

		MOPSO	NSGA-II	MODE
Fonction F1	Meilleur cas	8.25 $10^{-}5$	0.000129	0.298411
	Mauvais cas	0.000124	0.135470	0.442541
	Moyenne cas	**0.000101**	0.013578	0.007009
	Écart type	2.34 $10^{-}5$	0.364700	0.041130
Fonction F2	Meilleur cas	0.006450	0.005610	0. 006903
	Mauvais cas	0.008970	1.010000	0.010355
	Moyennecas	**0.007450**	0.028600	0.018558
	Écart type	0.000470	0.021200	0.001986
Fonction F3	Meilleur cas	0.000330	0.000600	0.000465
	Mauvais cas	0.152100	0.200100	0.183501
	Moyenne cas	**0.029900**	0.411000	0.043477
	Écart type	0.051200	0.052100	0.048121
Fonction F4	Meilleur cas	0.001200	0.002120	0.005120
	Mauvais cas	0.334000	0.538000	0.914650
	Moyenne cas	**0.024100**	0.042200	0.150770
	Écart type	0.101000	0.121400	0.213080

Cette partie est réservée pour une deuxième étude des approches d'optimisation proposées et mises en oeuvre et l'algorithme ε-MOPSO. Différemment de ce qu'on a utilisé

Table 3.3 – La métrique SP obtenu pour les fonctions de test.

		MOPSO	NSGA-II	MODE
Fonction F1	Meilleur cas	0.006670	0.000120	0.008514
	Mauvais cas	0.016120	0.013540	0.009274
	Moyenne cas	0.010110	**0.002870**	0.014000
	Écart type	0.001824	0.003684	0.003103
Fonction F2	Meilleur cas	0.057100	0.016700	0.058874
	Mauvais cas	0.112000	0.554000	0.088711
	Moyenne cas	0.051100	**0.025100**	0.056140
	Écart type	0.015400	0.015300	0.029923
Fonction F3	Meilleur cas	0.034500	0.021000	0.030267
	Mauvais cas	0.421000	0.032800	0.876420
	Moyenne cas	0.077100	**0.031200**	0.354121
	Écart type	0.102000	0.008510	0.240509
Fonction F4	Meilleur cas	0.036100	0.065400	0.001930
	Mauvais cas	0.502000	0.838000	1.414318
	Moyenne cas	0.092100	0.083100	**0.050052**
	Écart type	0.101000	0.315000	0.421240

comme concept de dominance, ce dernier est amélioré en utilisant la technique d'epsilon-dominance que nous venons de présenter. Le but de cette comparaison est d'étudier l'influence du mécanisme d'epsilon-dominance introduit sur les tests d'optimisation effectués. Cette étude est basée sur deux critères à savoir le temps de simulation et la diversité des solutions trouvées. Pour mesurer les performances des algorithmes cités ci-dessus, Mostaghim et Teich ont proposé une nouvelle métrique qui permet de mesurer la diversité des solutions d'une telle méthode d'optimisation multi-objectif [Mostaghim & Teich, 2003]. Cette métrique est inspirée de la méthode Sigma, dont le principe est de trouver le meilleur guide pour les particules dans l'essaim. La mise en oeuvre de cette technique a abouti aux résultats suggérés dans les tableaux 3.5, 3.6, 3.7 et 3.8. Il est à noter que les colonnes grisées de ces tableaux présentent les résultats trouvés pour les fronts de Pareto montrés dans les figures 3.12, 3.13, 3.14 et 3.15.

Pour les quatre tests effectués, la taille de l'archive, le temps de simulation et la diversité sont présentés dans les tableaux 3.5, 3.6, 3.7 et 3.8. Il est remarquable que l'augmentation

TABLE 3.4 – La métrique ER obtenu pour les fonctions de test.

		MOPSO	NSGA-II	MODE
Fonction F1	Meilleur	0.150000	0.010000	0.298411
	Mauvais	0.390000	0.971000	0.442541
	Moyenne	**0.286000**	0.373000	0.321000
	Écart type	0.088100	0.364700	0.251917
Fonction F2	Meilleur	0.130000	0.055000	0.020000
	Mauvais	0.323000	1.010000	1.045541
	Moyenne	**0.231700**	0.461000	0.242519
	Écart type	0.038800	0.377400	0.221670
Fonction F3	Meilleur	0.001000	0.010000	0.001103
	Mauvais	1.010000	1.010000	0.884540
	Moyenne	0.247000	0.384000	**0.225496**
	Écart type	0.383000	0.411000	0.331740
Fonction F4	Meilleur	0.071000	0.086000	0.733749
	Mauvais	0.190000	0.240000	1.101239
	Moyenne	**0.121000**	0.178000	0.926605
	Écart type	0.048800	0.067100	0.068739

de la taille de l'archive implique l'augmentation du temps de simulation. D'après ces tableaux, on peut déduire que pour les différents problèmes d'optimisation traités, le temps de simulation trouvé par l'algorithme ε-MOPSO est plus faible que celui pour les autres algorithmes ce qui prouve au mieux l'amélioration au niveau de la vitesse de convergence apportée par le mécanisme d'epsilon-dominance proposé. Pour les deux problèmes à deux objectifs, en comparant la diversité des solutions pour l'algorithme ε-MOPSO avec les autres algorithmes, on peut conclure que pour une taille élevée de l'archive, la valeur Div est la plus grande, ce qui signifie une meilleure diversité. Pour les deux problèmes à trois objectifs, contrairement aux premiers tests, les deux fonctions ont présenté des meilleures diversités pour les différents algorithmes. Cependant l'algorithme MODE donne une meilleure diversité des solutions non dominées que la méthode epsilon-dominance adoptée pour le problème F7.

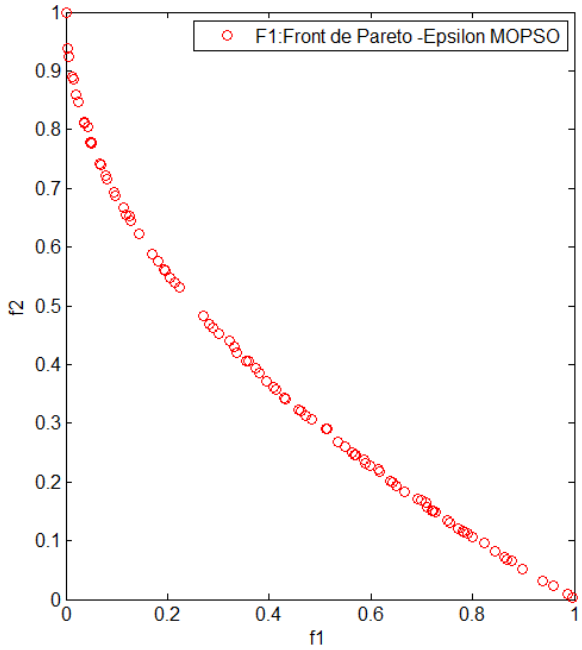

FIGURE 3.12 – Front de Pareto produit pour la fonction de test F1.

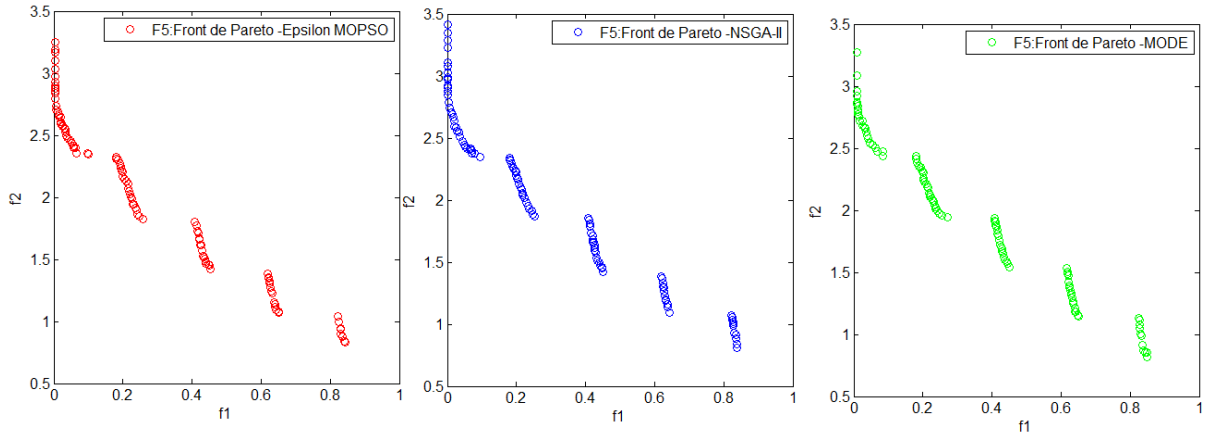

FIGURE 3.13 – Fronts de Pareto produits pour la fonction de test F5.

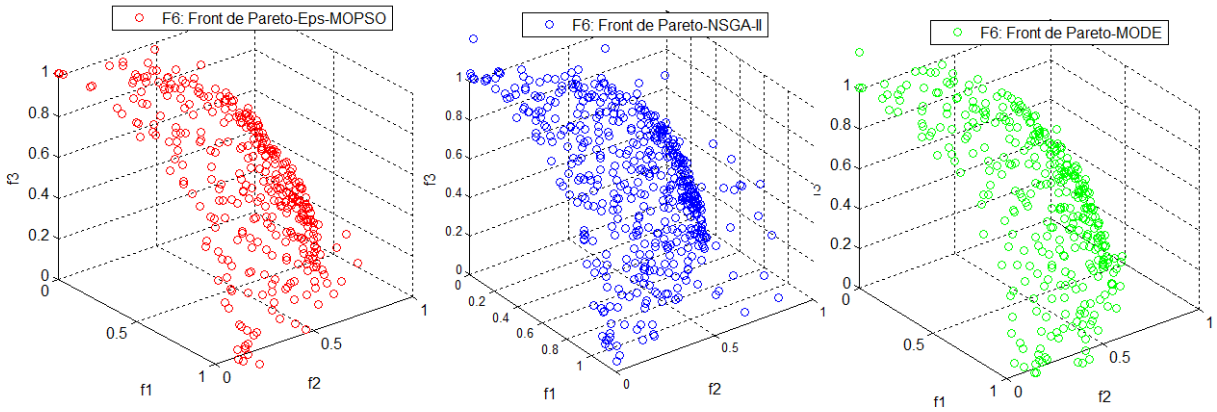

FIGURE 3.14 – Fronts de Pareto produits pour la fonction de test F6.

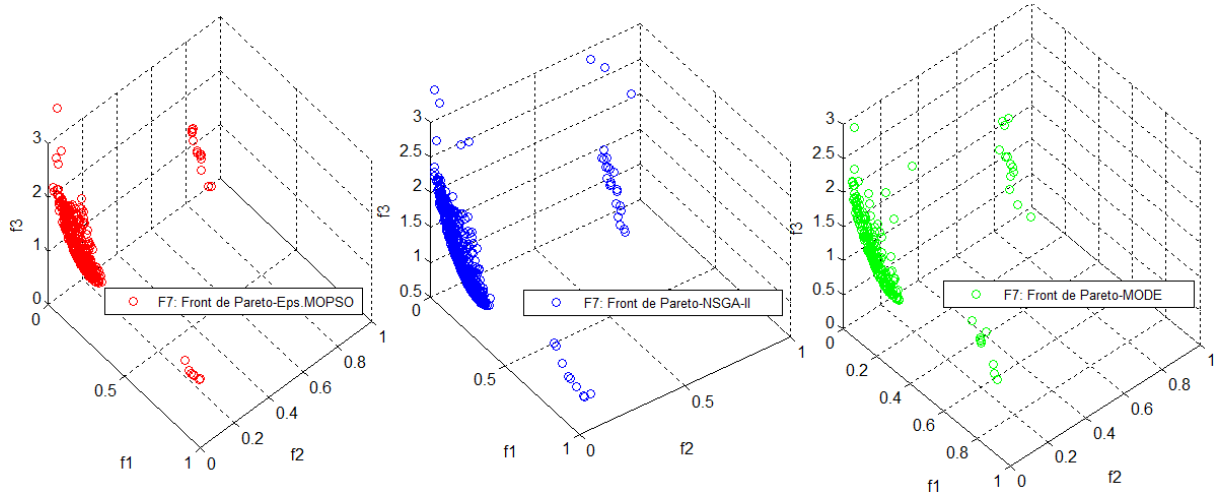

FIGURE 3.15 – Fronts de Pareto produits pour la fonction de test F7.

TABLE 3.5 – Résultats pour la fonction F1 à deux objectifs.

ε	0.05	0.025	0.02	0.01	0.001
Taille de l'archive	26	48	109	204	517
Temps de simulation (sec)ε-MOPSO	16,312	26,300	44,742	67,400	99,328
MODE				97,430	
NSGA-II				154,290	
Diversité (Div %) ε-MOPSO	0.804	0.933	0.772	0.930	0.931
MODE				0.871	
NSGA-II				0.642	

Table 3.6 – Résultats pour la fonction F5 à deux objectifs.

ε	0.005	0.0025	0.002	0.005	0.0001
Taille de l'archive	71	123	170	220	507
Temps de simulation (sec)ε-MOPSO	38,365	48,635	51,904	79,354	96,843
MODE				122,631	
NSGA-II				163,420	
Diversité (Div %) ε-MOPSO	0.34	0.44	0.65	0.59	0.55
MODE				0.58	
NSGA-II				0.55	

Table 3.7 – Résultats pour la fonction F6 à trois objectifs.

ε	0.06	0.05	0.04	0.03	0.015
Taille de l'archive	84	133	157	331	610
Temps de simulation (sec)ε-MOPSO	38,754	59,187	101,945	131,124	226,045
MODE				577,912	
NSGA-II				231,671	
Diversité (Div %) ε-MOPSO	0.86	0.91	0.705	0.923	0.982
MODE				0.921	
NSGA-II				0.754	

Table 3.8 – Résultats pour la fonction F7 à trois objectifs.

ε	0.04	0.03	0.025	0.020	0.015
Taille de l'archive	84	133	157	331	610
Temps de simulation (sec)ε-MOPSO	46,530	67,187	77,021	100,210	143.547
MODE					215,411
NSGA-II					353,202
Diversité (Div %) ε-MOPSO	0.638	0.725	0.715	0.880	0.895
MODE					0.927
NSGA-II					0.788

3.6 Conclusion

Dans ce chapitre, on a présenté l'algorithme d'optimisation par essaim particulaire MOPSO formulé dans un formalisme multi-objectif et préparé le formalisme à utiliser ultérieurement pour résoudre le problème de synthèse de la commande polynomiale RST. Un mécanisme d'epsilon-dominance est ensuite été introduit en vue d'améliorer les capacités de l'algorithme essentiellement en termes de rapidité de convergence et de diversité des solutions de Pareto obtenues.

Au début, on a présenté les concepts de base et le mode de fonctionnement de l'approche PSO dans un formalisme monobjectif avec une formulation mathématique de son pseudo-code. Ensuite, on a abordé l'adaptation et la formulation de cette technique, jugée robuste, performante et d'une supériorité remarquable par rapport à ses homologues connus en théorie d'optimisation par métaheuristiques, dans le formalisme multi-objectif. L'algorithme MOPSO ainsi proposé est amélioré en termes de diversité des solutions fournies et en rapidité moyennant le mécanisme d'epsilon-dominance introduit. Enfin, une mise en oeuvre pour l'optimisation d'un benchmark des fonctions de test issues de la littérature de l'optimisation multi-objectif est faite effectuée en vue de mettre en exergue les perfectionnements apportés à l'algorithme initialement proposé. Une étude comparative, des résultats obtenus, a été présentée et détaillée.

Le prochain chapitre portera sur la mise en oeuvre de l'outil MOPSO, original et perfectionné, d'optimisation multi-objectif pour la résolution des problèmes de synthèse et de réglage, bien reformulés pour différents processus dynamiques, des correcteurs numériques à structure canonique RST.

Chapitre 4

Synthèse de régulateurs polynomiaux RST par l'algorithme epsilon-MOPSO

4.1 Introduction

La synthèse des régulateurs numériques à structure canonique RST ainsi que le réglage de ses paramètres relèvent d'une complexité remarquable. Plusieurs travaux, déjà effectués, sont basés sur le choix, le plus souvent délicat et onéreux en temps de conception, des pôles à placer en boucle fermée. Ceci est toutefois indispensable afin de satisfaire les objectifs, indépendants et spécifiés dans les cahiers des charges, de régulation et de poursuite. Les fonctions de sensibilité permettent d'évaluer la stabilité et la robustesse de systèmes corrigés ainsi que la taille des incertitudes tolérées dans différentes régions du plan fréquentiel.

En présence des incertitudes du modèle, ces pôles ne peuvent pas être localisés de manière précise. La nécessité d'une approche systématique, permettant d'effectuer au mieux ce choix, se fait sentir dès qu'il s'agit de la commande de processus de plus en plus complexes. Un tel problème de commande robuste peut être formulé sous forme d'un problème d'optimisation multi-objectif et résolu en utilisant des métaheuristiques d'optimisation multi-objectif, en particulier les techniques MOPSO et epsilon-MOPSO proposées dans cette thèse.

Ce chapitre est organisé comme suit. Dans la première section, nous introduisons un aperçu général sur la théorie de correcteurs polynomiaux RST ainsi que les méthodes

classiques définies pour la synthèse. Nous détaillons plus particulièrement les approches par placement des pôles et de calibrage des fonctions de sensibilité. La notion des gabarits fréquentiels de robustesse est également présentée. La deuxième section sera réservée à la formulation du problème de synthèse directe des paramètres du correcteurs RST ainsi que la résolution moyennant les algorithmes MOPSO et epsilon-MOPSO proposés. Une mise en oeuvre pour la commande en vitesse variable d'un moteur DC est effectuée. La troisième et dernière partie sera consacrée au problème de synthèse RST moyennant une nouvelle formulation basée sur l'optimisation de fonctions de sensibilité désirées en boucle fermée. Une application à la commande en position d'un système de transmission flexible est présentée.

4.2 Synthèse de régulateurs numériques RST

4.2.1 Structure RST de régulateurs numériques

La figure 4.1 montre la structure canonique RST de régulation numérique de processus discrets [Landau, 1998 ; Landau et Karimi, 1998]. Le modèle discret du processus, mono-entrée-mono sortie, est décrit sous la forme suivante :

$$G\left(z^{-1}\right) = z^{-d}\frac{B\left(z^{-1}\right)}{A\left(z^{-1}\right)} \tag{4.1}$$

avec d est le retard pur du modèle, exprimé en un nombre fini des pas d'échantillonnage T_s, z^{-1} étant l'opérateur de retard, i.e. $Z\left(y\left(t-1\right)\right) = z^{-1}Z\left(y\left(t\right)\right)$. Les signaux $p\left(t\right)$ et $b\left(t\right)$ représentent respectivement les perturbations et les bruits sur la sortie du système.

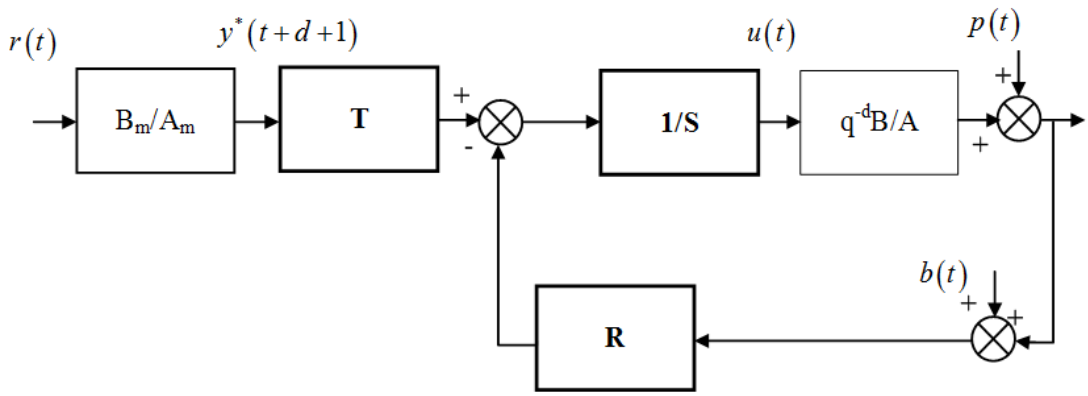

FIGURE 4.1 – Boucle de régulation numérique avec un correcteur RST.

Les polynômes A et$B$, supposés premiers entre eux et qui représentent respectivement le numérateur et le dénominateur du système, sont définis comme suit :

$$\begin{cases} B\left(z^{-1}\right) = b_1 z^{-1} + b_2 z^{-2} + ... + b_{n_B} z^{-n_B} \\ A\left(z^{-1}\right) = 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_{n_A} z^{-n_A} \end{cases} \tag{4.2}$$

Les dynamiques, indépendantes, de régulation et de poursuite de cette structure de commande numérique sont définies par les trois polynômes R, S et T du régulateur RST à deux degrés de liberté. En effet, la dynamique de régulation en termes de stabilité et de rejet de perturbation est imposée par les pôles de la boucle fermée moyennant les polynômes R et S. Les performances en poursuite sont définies par le polynôme T. Ces trois polynômes, inconnues du problème de synthèse RST, sont définis comme suit :

$$T\left(z^{-1}\right) = t_0 + t_1 z^{-1} + t_2 z^{-2} + ... + t_{n_T} z^{-n_T} \tag{4.3}$$

$$R\left(z^{-1}\right) = r_0 + r_1 z^{-1} + r_2 z^{-2} + ... + r_{n_R} z^{-n_R} \tag{4.4}$$

$$S\left(z^{-1}\right) = s_0 + s_1 z^{-1} + s_2 z^{-2} + ... + s_{n_S} z^{-n_S} \tag{4.5}$$

avec $n_A = \deg(A)$, $n_B = \deg(B)$, $n_R = \deg(R)$, $n_S = \deg(S)$, $n_T = \deg(T)$.

Dans le domaine temporel, la loi de commande numérique RST à appliquer au procédé est présentée par l'équation suivante :

$$S\left(z^{-1}\right) u\left(t\right) = T\left(z^{-1}\right) y^*\left(t+d+1\right) - R\left(z^{-1}\right) y\left(t\right) \tag{4.6}$$

Le terme $y^*\left(t+d+1\right)$ dans la loi de commande RST de l'équation (4.6) représente la trajectoire de référence du système corrigé. Cette consigne de référence peut être enregistrée dans un microcontrôleur ou engendrée à partir d'un modèle de référence de la forme [Landau, 2002] :

$$G_{ref}\left(z^{-1}\right) = \frac{B_m\left(z^{-1}\right)}{A_m\left(z^{-1}\right)} \tag{4.7}$$

Dans le cas de la commande RST avec modèle de référence, cette trajectoire est définie en fonction du signal d'entrée $r\left(t\right)$ comme suit :

$$y^*\left(t+d+1\right) = \frac{B_m\left(z^{-1}\right)}{A_m\left(z^{-1}\right)} r\left(t\right) \tag{4.8}$$

La fonction de transfert de la boucle fermée entre la sortie $y\left(t\right)$ du système et la référence $y^*\left(t+d+1\right)$ est alors définie par :

$$H_{BF}\left(z^{-1}\right) = \frac{z^{-d} B\left(z^{-1}\right) T\left(z^{-1}\right)}{P\left(z^{-1}\right)} \tag{4.9}$$

avec le polynôme $P\left(z^{-1}\right) = P_D\left(z^{-1}\right) P_F\left(z^{-1}\right)$, dont les racines représentent les pôles dominants et auxiliaires de la boucle fermée, est défini par l'expression suivante :

$$P\left(z^{-1}\right) = A\left(z^{-1}\right) S\left(z^{-1}\right) + z^{-d} B\left(z^{-1}\right) R\left(z^{-1}\right) = P_D\left(z^{-1}\right) P_F\left(z^{-1}\right) \tag{4.10}$$

4.2.2 Approche de synthèse par placement de pôles

Le principe de cette méthode de synthèse des régulateurs RST, proposée à l'origine par I.D. Landau, voir [Landau, 1998], consiste à imposer les pôles, dominants et auxiliaires, de la boucle fermée du système qui revient en fait à résoudre une équation polynomiale à deux inconnus $R\left(z^{-1}\right)$ et $S\left(z^{-1}\right)$. Cette équation polynomiale, dite de Bézout, est donnée par l'équation (4.11). Le polynôme $T\left(z^{-1}\right)$ est calculé, indépendamment, selon qu'il s'agit d'un problème de régulation ou de poursuite [Landau et al., 1998].

$$P\left(z^{-1}\right) = A\left(z^{-1}\right) S\left(z^{-1}\right) + z^{-d} B\left(z^{-1}\right) R\left(z^{-1}\right) = p_0 + p_1 z^{-1} + p_2 z^{-1} + \ldots + p_{n_P} z^{-1} \tag{4.11}$$

L'équation diophantienne (4.11), dont la résolution fait l'objet de la synthèse RST, est une équation polynomiale dans la quelle les polynômes A, B et P sont connus. Les coefficients des polynômes $R\left(z^{-1}\right)$ et $S\left(z^{-1}\right)$, inconnues du problème de synthèse RST, peuvent être calculés moyennant l'écriture matricielle (4.12) dite de Sylvester.

$$\underbrace{\begin{pmatrix} a_0 & 0 & \cdots & 0 & b_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \ddots & \vdots & b_1 & b_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & 0 \\ a_{n_A} & \vdots & \ddots & a_0 & b_{n_B} & \vdots & \ddots & b_0 \\ 0 & \ddots & \vdots & a_1 & 0 & \ddots & \vdots & b_1 \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n_A} & 0 & \cdots & 0 & b_{n_B} \end{pmatrix}}_{\mathcal{M}} \underbrace{\begin{pmatrix} s_0 \\ \vdots \\ s_{n_S} \\ r_0 \\ \vdots \\ \vdots \\ r_{n_R} \end{pmatrix}}_{\mathcal{X}} = \underbrace{\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n_P} \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{\mathcal{P}} \tag{4.12}$$

La résolution du système d'équations linéaire ainsi obtenus permettra de déterminer les inconnus $(r_i)_{0 \leq i \leq n_R}$ et $(s_i)_{0 \leq i \leq n_S}$ de la loi de commande RST comme suit :

$$\mathcal{X} = \mathcal{M}^{-1} \mathcal{P} \tag{4.13}$$

avec $\mathcal{M}$ la matrice carrée de Sylvester de dimension $(n_A + n_B) \times (n_A + n_B)$. L'équation

de Bézout (4.11) a une solution unique de degré minimale sous les conditions suivantes :

$$\begin{cases} \deg\left(P\left(z^{-1}\right)\right) = n_P \leq n_A + n_B + d - 1 \\ \deg\left(S\left(z^{-1}\right)\right) = n_s = n_B + d - 1 \\ \deg\left(R\left(z^{-1}\right)\right) = n_R = n_A - 1 \end{cases} \tag{4.14}$$

4.2.3 Approche par placement de pôles et calibrage des fonctions de sensibilité

4.2.3.1 Principe de base

L'objectif principal de l'approche de synthèse RST par placement de pôles et calibrage des fonctions de sensibilité est de satisfaire au mieux les spécifications souhaitées du cahier des charges en termes de stabilité robuste et de performances nominales en poursuite et en régulation. Ceci est achevé moyennant l'introduction des parties fixes pré-spécifiées dans les polynômes $R\left(z^{-1}\right)$ et $S\left(z^{-1}\right)$ du correcteur RST ainsi que le calibrage itératif, dit aussi *shaping*, de certaines fonctions de sensibilité de la boucle fermée [Landau, 1998 ; Landau et al., 1998].

Les fonctions de sensibilité présentent un outil performant pour l'étude des propriétés de la boucle fermée de processus discrets en termes de robustesse, à savoir les marges de stabilité, ainsi que les performances nominales de la boucle de régulation dans le plan fréquentiel. Ces spécifications de robustesse et de performances nominales sont traduites, dans ce cadre, par des gabarits fréquentiels dits de robustesse, Figure 4.2, sur les modules des fonctions de sensibilité définies ultérieurement.

Cette approche, à caractère itératif, de synthèse de correcteurs polynomiaux RST est résumée à travers les étapes suivantes [Landau & Karimi, 1998] :

i. Placement des pôles dominants de la boucle fermée issu généralement du choix d'une dynamique équivalente d'un système du deuxième ordre décrit par sa pulsation propre et son facteur d'amortissement.

ii. Introduction des parties fixes pré-spécifiées $H_R\left(z^{-1}\right)$ et $H_S\left(z^{-1}\right)$ respectivement dans les polynômes $R\left(z^{-1}\right)$ et $S\left(z^{-1}\right)$ du régulateur RST afin d'assurer le rejet des perturbations externes sur la sortie du système et l'atténuation des bruits de mesure en haute fréquences.

iii. Calcul des polynômes $R\left(z^{-1}\right)$ et $S\left(z^{-1}\right)$ à partir de l'identité de Bézout obtenue du système augmenté des filtres de pondération $H_R\left(z^{-1}\right)$ et $H_S\left(z^{-1}\right)$ introduits.

iv. Vérifier les réponses fréquentielles des fonctions de sensibilité, en particulier celle dite de sortie, si elles vérifient les limites supérieures des gabarits de robustesse définis. Si c'est le cas alors la procédure de synthèse RST est achevée, sinon procéder au calibrage des fonctions de sensibilité mises en jeu par la modification du placement des pôles dominants, l'ajout des pôles auxiliaires ou la ré-définition des parties fixes définies à l'étape ii).

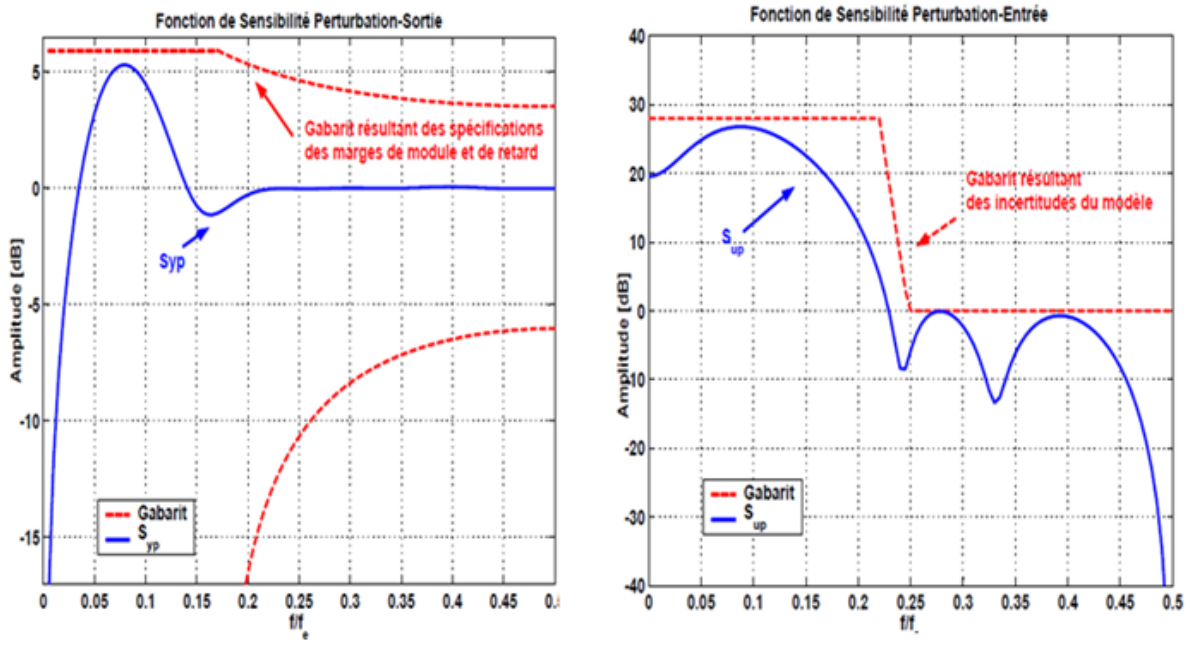

FIGURE 4.2 – Formes générales des gabarits sur les fonctions de sensibilité [Zito, 2005].

4.2.3.2 Fonctions de sensibilité

La méthode de synthèse RST par placement de pôles et calibrage de fonctions de sensibilité considère principalement les signaux externes $b(t)$ et $p(t)$ qui représentent respectivement les bruits de mesure en hautes fréquences et les perturbations additives sur la sortie du procédé en boucle fermée. L'influence de ces signaux exogènes sur l'entrée de commande $u(t)$ et la sortie $y(t)$ du système est bien évaluée et interprétée moyennant les fonctions de sensibilité définies dans les équations (4.15), (4.16) et (4.17) :

– Fonction de sensibilité perturbation-sortie :

$$S_{yp}\left(z^{-1}\right) = \frac{A\left(z^{-1}\right)S\left(z^{-1}\right)}{A\left(z^{-1}\right)S\left(z^{-1}\right) + z^{-d}B\left(z^{-1}\right)R\left(z^{-1}\right)} = \frac{A\left(z^{-1}\right)S\left(z^{-1}\right)}{P\left(z^{-1}\right)} \tag{4.15}$$

– Fonction de sensibilité perturbation-entrée :

$$S_{up}\left(z^{-1}\right) = \frac{-A\left(z^{-1}\right)R\left(z^{-1}\right)}{A\left(z^{-1}\right)S\left(z^{-1}\right) + z^{-d}B\left(z^{-1}\right)R\left(z^{-1}\right)} = \frac{-A\left(z^{-1}\right)R\left(z^{-1}\right)}{P\left(z^{-1}\right)} \tag{4.16}$$

– Fonction de sensibilité bruit-sortie :

$$S_{yb}\left(z^{-1}\right) = \frac{-z^{-d}B\left(z^{-1}\right)R\left(z^{-1}\right)}{A\left(z^{-1}\right)S\left(z^{-1}\right) + z^{-d}B\left(z^{-1}\right)R\left(z^{-1}\right)} = \frac{-z^{-1}B\left(z^{-1}\right)R\left(z^{-1}\right)}{P\left(z^{-1}\right)} \tag{4.17}$$

4.2.3.3 Calcul de la dynamique de régulation

Dans cette approche de synthèse des correcteurs RST, les polynômes R et S sont paramétrés moyennant les parties fixes pré-spécifiées de la façon suivante :

$$R\left(z^{-1}\right) = R'\left(z^{-1}\right)H_R\left(z^{-1}\right) \tag{4.18}$$

$$S\left(z^{-1}\right) = S'\left(z^{-1}\right)H_S\left(z^{-1}\right) \tag{4.19}$$

A titre d'exemple, le rejet de perturbations constantes sur la sortie de système peut être garanti en spécifiant un terme intégrateur dans la chaîne d'action, c'est-à-dire dans l'expression du polynôme $S\left(z^{-1}\right)$. Ceci est achevé en choisissant $H_S\left(z^{-1}\right) = 1 - z^{-1}$. Avec cette spécification des parties fixes au niveau des polynômes du régulateur RST, le polynôme caractéristique permettant d'imposer les pôles de la boucle fermée du système corrigé est alors donné par :

$$P\left(z^{-1}\right) = A\left(z^{-1}\right)S'\left(z^{-1}\right)H_S\left(z^{-1}\right) + z^{-d}B\left(z^{-1}\right)R'\left(z^{-1}\right)H_R\left(z^{-1}\right) \tag{4.20}$$

Soit de la nouvelle forme de l'équation de Diophante :

$$P\left(z^{-1}\right) = A'\left(z^{-1}\right)S'\left(z^{-1}\right) + z^{-d}B'\left(z^{-1}\right)R'\left(z^{-1}\right) \tag{4.21}$$

avec $A'\left(z^{-1}\right) = A\left(z^{-1}\right)H_S\left(z^{-1}\right)$ et $B'\left(z^{-1}\right) = B\left(z^{-1}\right)H_R\left(z^{-1}\right)$ le numérateur et le dénominateur du système augmenté. La résolution de l'équation (4.21) ainsi que les conditions sur les degrés des différents polynômes de la boucle de régulation sont déduites de la même manière que dans le cas classique de synthèse RST via la technique de Sylvester, Equations (4.12), (4.13) et (4.14).

4.2.3.4 Calcul de la dynamique de poursuite

Le polynôme $T\left(z^{-1}\right)$, relatif à la dynamique de poursuite du correcteur RST, est calculé indépendamment des polynômes de régulation $R\left(z^{-1}\right)$ et $S\left(z^{-1}\right)$, d'où l'intérêt apporté

à ce type des lois de commande à deux degrés de liberté (2-DOF). Différentes techniques sont proposées dans la littérature pour le calcul de ce polynôme. Nous présentons, selon qu'il s'agisse d'un problème de régulation ou un problème de poursuite, les choix suivants [Landau, 1998] :

1. $T\left(z^{-1}\right) = \frac{P\left(z^{-1}\right)}{B(1)}$, permet d'imposer comme dynamique de poursuite celle du modèle de référence G_{ref} ainsi que la possibilité de simplification des pôles imposés par la boucle de régulation,
2. $T\left(z^{-1}\right) = \frac{P_D\left(z^{-1}\right)}{B(1)P_F(1)}$, correspond à simplifier les seuls pôles dominants de la régulation en laissant inchangés les pôles auxiliaires,
3. $T\left(z^{-1}\right) = t_0 = \frac{P(1)}{B(1)}$, s'il s'agit juste d'un problème de régulation plutôt que de poursuite.

4.3 Synthèse RST par la technique epsilon-MOPSO proposée

Dans cette section, la synthèse d'un régulateur numérique à structure RST est formulée sous forme d'un problème d'optimisation multi-objectif sous contraintes. La résolution d'un tel problème non linéaire est assurée via l'algorithme perfectionné epsilon-MOPSO.

4.3.1 Critères de performance et contraintes pour la synthèse

4.3.1.1 Critères de performances temporels

Les plus classiques sont liés essentiellement aux caractéristiques de la réponse indicielle du système corrigé, telles que l'erreur statique, le temps de montée, le temps d'établissement et le dépassement. Une autre alternative consiste à utiliser d'autres types de quanti ?cateurs, issus de la théorie de la commande optimale. Les critères d'optimalité correspondants peuvent avoir des formes variées dont les plus répandues en pratique sont définies comme suit :

– Critère IAE :

$$J_{IAE}\left(\boldsymbol{\theta}\right) = \int_0^{+\infty} \left|e\left(\boldsymbol{\theta}, t\right)\right| dt \tag{4.22}$$

– Critère ITAE :

$$J_{ITAE}\left(\boldsymbol{\theta}\right) = \int_0^{+\infty} t\left|e\left(\boldsymbol{\theta}, t\right)\right| dt \tag{4.23}$$

– Critère ISE :

$$J_{ISE}(\boldsymbol{\theta}) = \int_0^{+\infty} e(\boldsymbol{\theta}, t)^2 \, dt \tag{4.24}$$

– Critère ITSE :

$$J_{ITSE}(\boldsymbol{\theta}) = \int_0^{+\infty} te(\boldsymbol{\theta}, t)^2 \, dt \tag{4.25}$$

– Critère EITSE :

$$J_{EITSE}(\boldsymbol{\theta}) = \int_0^{+\infty} t^p e(\boldsymbol{\theta}, t)^2 \, dt \ , p \geq 0 \tag{4.26}$$

– Critère MO :

$$J_{MO}(\boldsymbol{\theta}, t) = \frac{y_{\max}(\boldsymbol{\theta}, t) - y(\boldsymbol{\theta}, t \to +\infty)}{y(\boldsymbol{\theta}, t \to +\infty)} \tag{4.27}$$

4.3.1.2 Critères de performances fréquentiels

Les plus classiques sont liés aux caractéristiques de la réponse fréquentielle du système corrigé, soit dans le plan de Bode comme la bande passante et le facteur de résonnance, soit dans le plan de Nyquist telles que les marges de stabilité. Comme pour le cas temporel, ces contraintes peuvent définir des gabarits fréquentiels de performance. Des formes similaires à celle de la figure 4.2 peuvent être retenues pour la synthèse et le réglage de correcteurs RST. Dans la littérature, d'autres indices de performance, plus adaptés pour des problèmes synthèse issue de la théorie de commande optimale, peuvent être utilisés. Nous soulignons essentiellement l'égalité de Zames-Francis adoptée ultérieurement [Zames & Francis, 1983 ; Francis & Zames, 1984].

4.3.1.3 Contraintes de robustesse et de performance nominale

Pour des raisons de robustesse et de performance nominale telles que le rejet de perturbations constantes et l'atténuation des bruits de hautes fréquence, nous introduisons dans les polynômes $R(z^{-1})$ et $S(z^{-1})$ les parties fixes mentionnées dans les équations (4.18) et (4.19). Conformément aux interprétations menées par Landau et Karimi [Landau & Karimi, 1998] sur les formes choisies pour ces polynômes, nous retenons les expressions suivantes : (4.28)

$$H_S\left(z^{-1}\right) = 1 - z^{-1} \tag{4.28}$$

$$H_R\left(z^{-1}\right) = 1 + z^{-1} \tag{4.29}$$

Afin d'assurer une marge de retard $\Delta\tau = T_s$ ainsi qu'une marge de module $\Delta M \geq 0.5$, le module de la fonction de sensibilité de sortie $S_{yp}(z^{-1})$ se voit contraint par les limites

supérieurs et inférieurs du gabarit fréquentiel de robustesse conformément aux inégalités suivantes :

$$\left|W^{-1}\right|_{\inf} = 1 - \left|1 - z^{-1}\right| \leq \left|S_{yp}\left(z^{-1}\right)\right| \tag{4.30}$$

$$\left|W^{-1}\right|_{\sup} = 1 + \left|1 - z^{-1}\right| \geq \left|S_{yp}\left(z^{-1}\right)\right| \tag{4.31}$$

Dans les équations (4.30) et (4.31), les filtres de pondération $|W^{-1}|_{\sup}$ et $|W^{-1}|_{\inf}$ représentent respectivement les limites supérieure et inférieure de robustesse dans le domaine de fréquentiel. Ces deux expressions analytiques représenteront dans ce qui suit les contraintes de type inégalités non linaires pour le problème de synthèse RST formulé comme un problème d'optimisation multi-objectif.

4.3.2 Formulation du problème d'optimisation

Le problème synthèse RST, vu comme un problème d'optimisation multi-objectif, peut être ramené à un problème de recherche des variables de décision optimales $\boldsymbol{\theta}^* = [x_1^*,\, x_2^*,\, ...,\, x_m^*]^T$ qui représentent les coefficients $(r_i)_{0\leq i\leq n_R}$, $(s_i)_{0\leq i\leq n_S}$ voire $(t_i)_{0\leq i\leq n_T}$ des polynômes formants la structure de contrôle RST. Pour le cas de régulation, ces paramètres, inconnus du problème, sont regroupés dans le vecteur de décision donné comme suit :

$$\boldsymbol{\theta} = [s_0,\, s_1,\, ...,\, s_{n_S}, r_0,\, r_1,\, ...,\, r_{n_R}]^T \tag{4.32}$$

Pour un problème bi-objectif, ces variables de décision minimisent les deux critères de performance choisis ici comme le dépassement maximal de la réponse indicielle (MO) et l'indice sur l'erreur en régime permanent (IAE), donnés respectivement comme suit :

$$f_{MO}\left(\boldsymbol{\theta}, t\right) = \frac{y_{\max}\left(\boldsymbol{\theta}, t\right) - y\left(\boldsymbol{\theta}, t \to +\infty\right)}{y\left(\boldsymbol{\theta}, t \to +\infty\right)} \tag{4.33}$$

$$f_{IAE}\left(\boldsymbol{\theta}, t\right) = \int_0^{+\infty} \left|\varepsilon\left(\boldsymbol{\theta}, t\right)\right| dt \tag{4.34}$$

Dans cette conception, on note que seule la dynamique de régulation est considérée, c'est-à-dire nous nous intéressons uniquement au calcul des polynômes $S\left(z^{-1}\right)$ et $R\left(z^{-1}\right)$ du régulateur polynomial RST. Par conséquent, le polynôme $T\left(z^{-1}\right)$ peut être déterminé systématiquement de la façon suivante :

$$T\left(z^{-1}\right) = t_0 = \frac{P\left(1\right)}{B\left(1\right)} \tag{4.35}$$

En effet, nous avons choisi de réduire la taille du problème d'optimisation formulé pour la synthèse RST. Seule la dynamique de régulation en boucle fermée est envisagée pour cette

étude de cas. Rien n'empêche, évidement, à traiter aussi bien le problème de poursuite en plus. Dans ce cas, le problème d'optimisation devient de taille plus grande et d'autres variables décision, de type coefficients $(t_i)_{0\leq i\leq n_T}$ du polynôme de poursuite $T\left(z^{-1}\right)$, seront ajoutées dans le vecteur $\boldsymbol{\theta}$ de l'équation (4.32).

4.3.3 Mise en oeuvre pour la commande en vitesse d'un moteur DC

4.3.3.1 Description du procédé

Le processus considéré dans cet exemple de mise en oeuvre est un banc de tests relatif à une machine à courant continu commandée en vitesse. Cette machine est de type excitation séparée, de puissance 250 watts et de vitesse nominale de rotation de 3000 tr/mn. Ce moteur DC est alimenté par un convertisseur statique de type AC-DC. La modélisation de l'ensemble convertisseur-machine-capteur considéré, sous forme d'un système du second ordre décrit par l'équation (4.36). Cette modélisation, avec un gain statique G_0, des constantes de temps mécanique τ_m et électrique τ_e, a abouti par application des méthodes d'identification expérimentales [Ayadi et al., 2008 ; Bouallègue et al., 2012], aux résultats numériques rassemblés dans le tableau 4.1.

$$G\left(s\right)=\frac{G_0}{\left(1+\tau_m s\right)\left(1+\tau_e s\right)} \tag{4.36}$$

Ce modèle du processus, avec les valeurs nominales de ses paramètres suggérés dans le tableau 4.1, est discrétisé en adoptant une période d'échantillonnage $T_s = 10$ ms.

Table 4.1 – Paramètres identifiés du moteur DC.

Paramètres	Valeurs nominales	Incertitude paramétrique
G_0	0.05	$\pm 50\%$
τ_m	300 ms	$\pm 50\%$
τ_e	14 ms	$\pm 50\%$

4.3.3.2 Résultats de simulation et discussion

Dans cette étude de cas, le problème de synthèse d'un correcteur digital RST pour la commande à vitesse variable du moteur DC considéré est posé sous forme du problème

d'optimisation multi-objectif contraint suivant [Madiouni et al., 2016] :

$$\underset{\substack{\boldsymbol{\theta}=[s_1,\,r_0,r_1]^T\in\Re^3 \\ \text{sous contraintes:}}}{\text{minimiser}} \quad \begin{array}{c}(f_1(\boldsymbol{\theta}),\,f_2(\boldsymbol{\theta})) \\ g_1(\boldsymbol{\theta})=|S_{yp}(z^{-1},\,\boldsymbol{\theta})|-|W^{-1}|_{\sup}\leq 0 \\ g_2(\boldsymbol{\theta})=|W^{-1}|_{\inf}-|S_{yp}(z^{-1},\,\boldsymbol{\theta})|\leq 0\end{array} \tag{4.37}$$

avec $g_1(\boldsymbol{\theta})$ et $g_2(\boldsymbol{\theta})$ sont les contraintes non linéaire de type inégalité du problème formulé, $\boldsymbol{\theta} = [s_1, r_0, r_1]^T \in \Re^3$ étant le vecteur des variables de décision, inconnu du problème, $f_1(\boldsymbol{\theta})$ et $f_2(\boldsymbol{\theta})$ sont les deux objectifs MO et IAE à minimiser choisis conformément aux équations (4.33) et (4.34).

On rappelle que la fonction de sensibilité de sortie $|S_{yp}(z^{-1},\,\boldsymbol{\theta})|$ de l'équation (4.37) représente la fonction de transfert en boucle fermée entre la sortie du procédé $y(t)$ et l'entrée de perturbation constante $p(t)$. Cette sensibilité est définie en fonction des paramètres de décision du problème comme suit :

$$S_{yp}\left(z^{-1},\,\boldsymbol{\theta}\right) = \frac{A\left(z^{-1}\right)S\left(z^{-1},\boldsymbol{\theta}\right)}{A\left(z^{-1}\right)S\left(z^{-1},\,\boldsymbol{\theta}\right)+z^{-d}B\left(z^{-1}\right)R\left(z^{-1},\,\boldsymbol{\theta}\right)} \tag{4.38}$$

La résolution du problème de synthèse multi-objectif RST de l'équation (4.37) est accomplie moyennant les algorithmes métaheuristiques MODE, NSGA-II, MOPSO et epsilon-MOPSO, étudiés et perfectionnés dans les chapitres précédents. Les résultats d'optimisation relatifs aux 20 exécutions des différents algorithmes sur ce problème contraint de synthèse sont donnés dans le tableau 4.2, pour les variables de décision et le tableau 4.3 pour le temps de calcul de différents algorithmes.

Table 4.2 – Résultats d'optimisation du problème (4.37), cas moyen.

Algorithmes	s_1^*	r_0^*	r_1^*
epsilon-MOPSO	0.511	0.046	0.443
MOPSO	0.504	0.044	0.455
NSGA-II	0.371	0.038	0.461
MODE	0.571	0.041	0.399

Table 4.3 – Temps de calcul relatif aux algorithmes d'optimisation proposés.

Algorithmes	epsilon-MOPSO	NSGA-II	MOPSO	MODE
Temps de calcul(en seconde)	95.291	132.103	108.351	128.079

Les coefficients optimisés $(r_i^*)_{0\leq i\leq n_R}$ et $(s_i^*)_{0\leq i\leq n_S}$ du régulateur digital RST du tableau 4.2 sont obtenus dans le cas moyen d'optimisation. On note aussi que toutes les solutions du front de Pareto obtenues en rapport avec les objectifs f_1 et f_2 sont non-dominées et peuvent être considérées comme des solutions possibles pour le problème d'optimisation considéré. Ceci peut être observé à travers les figures 4.3 et 4.4. Nous choisissons, dans ce cas, arbitrairement une solution quelconque de la surface de compromis comme un optimum du problème d'optimisation multi-objectif formulé. D'autre part et afin de confirmer

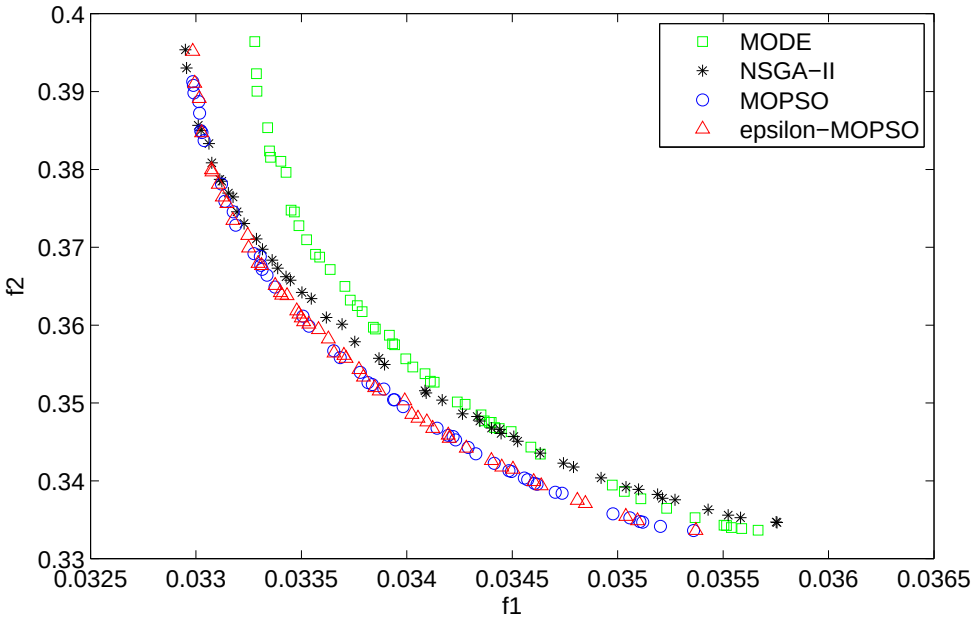

FIGURE 4.3 – Fronts de Pareto des algorithmes pour la résolution du problème (4.37).

cette répartition des solutions sur les fronts de Pareto du problème, nous avons exécuté l'algorithme perfectionné epsilon-MOPSO pour différentes valeurs de ses paramètres de contrôle, en particulier le facteur d'inertie. Les résultats obtenus sont présentés dans la figure 4.4. Ces résultats montrent la robustesse de la technique proposée vis-à-vis de la variation des paramètres de son algorithme. Ceci justifie davantage le recours à cette méthode d'optimisation globale.

De point de vue dynamique de convergence, les algorithmes métaheuristiques proposés sont comparés entre eux lors de la recherche des coefficients des polynômes $R(z^{-1},\boldsymbol{\theta})$ et $S(z^{-1},\boldsymbol{\theta})$ du correcteur RST multi-objectif aussi bien pour le critère MO que le critère IAE. Les figures 4.5 et 4.6 montre cette comparaison qui met en exergue la supériorité de l'algorithme epsilon-MOPSO par rapport à ses homologues.

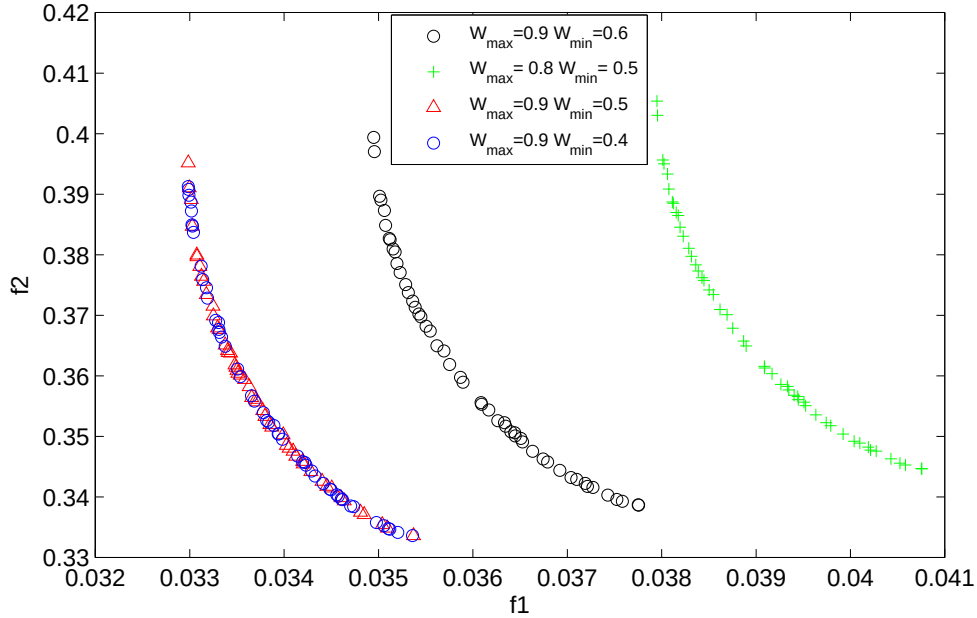

FIGURE 4.4 – Robustesse de l'algorithme epsilon-MOPSO vis-à-vis la variation paramétrique du facteur d'inertie.

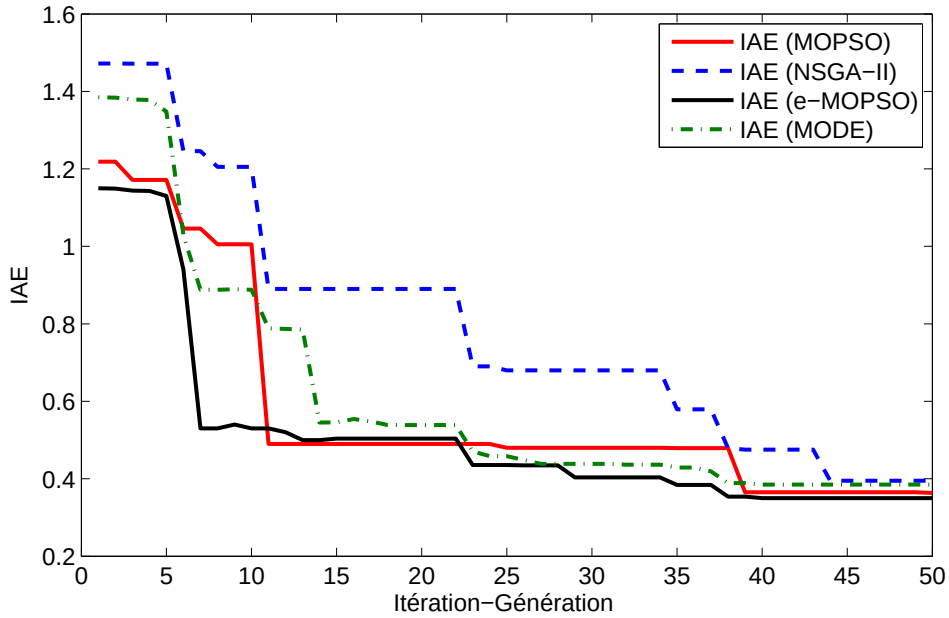

FIGURE 4.5 – Dynamique de convergence des algorithmes pour l'objectif $f_1(\boldsymbol{\theta})$

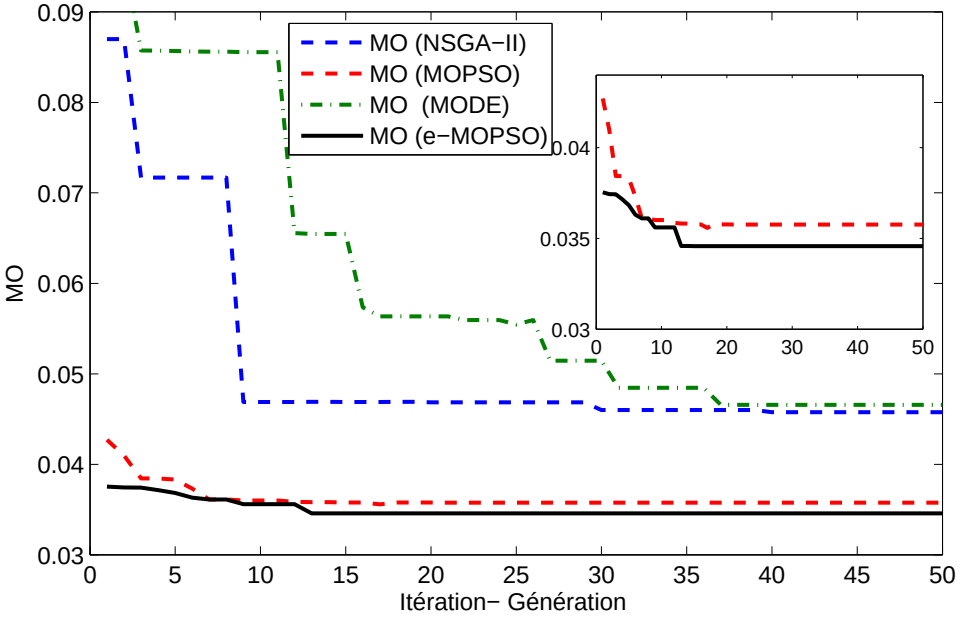

FIGURE 4.6 – Dynamique de convergence des algorithmes pour l'objectif $f_2(\boldsymbol{\theta})$

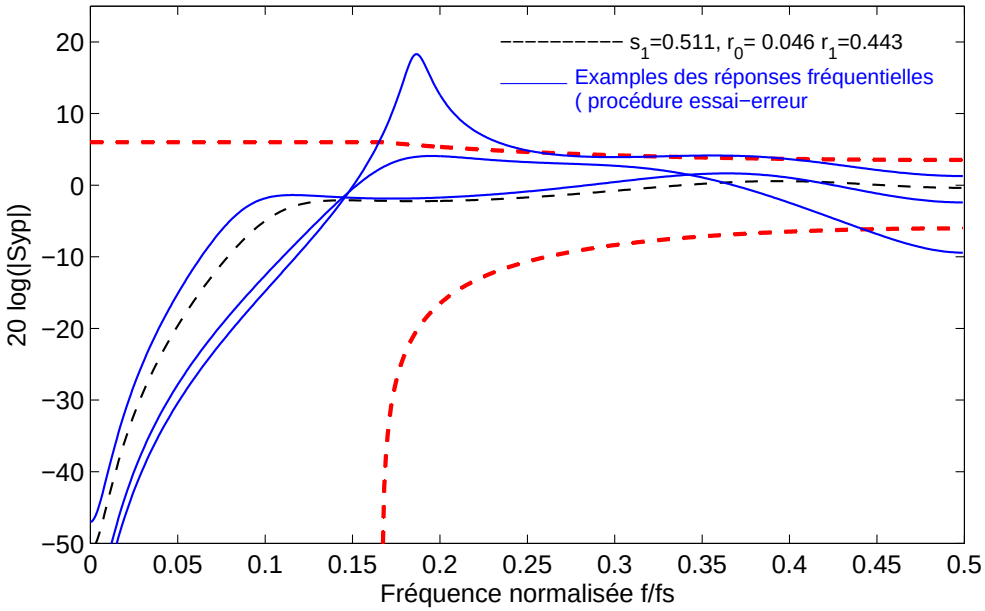

FIGURE 4.7 – Réponse fréquentielle de la fonction de sensibilité de sortie $S_{yp}(\boldsymbol{\theta}^*)$

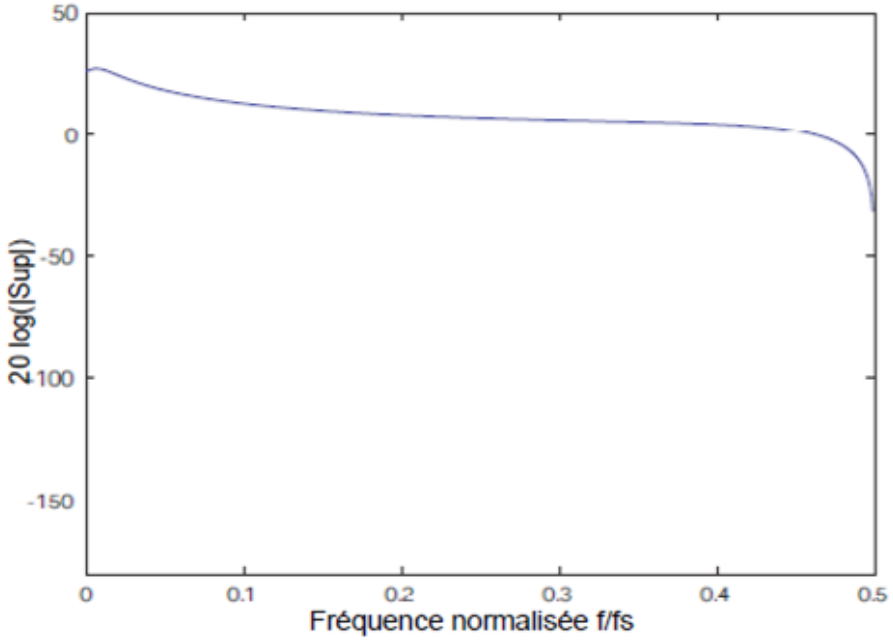

FIGURE 4.8 – Réponse fréquentielle de la fonction de sensibilité d'entrée $S_{up}(\boldsymbol{\theta}^*)$

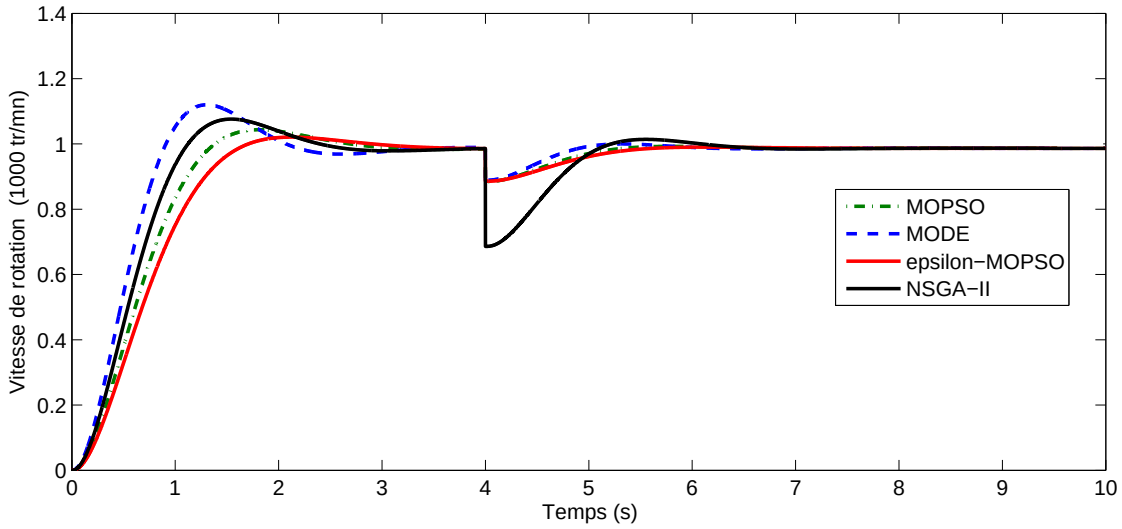

FIGURE 4.9 – Réponses indicielles du système en boucle fermée.

Afin d'analyser les propriétés de robustesse du correcteur RST synthétisé par approches métaheuristiques proposées, nous présentons quelques réponses fréquentielles de fonctions de sensibilité de sortie et d'entrée, respectivement, dans les figures 4.7 et 4.8. Le module de la fonction de sensibilité de sortie $S_{yp}(\boldsymbol{\theta})$ reste à l'intérieur du gabarit de robustesse prédéfini. Pour la fonction de sensibilité d'entrée $S_{up}(\boldsymbol{\theta})$, l'atténuation en haute fréquences de son module vérifie bien l'atténuation assurée des bruits de mesure. Ces résultats conduisent également à l'obtention de bonnes performances dans le domaine temporel du régulateur RST conçu et optimisé à base des algorithmes métaheuristiques proposés, comme montrés dans la figure 4.9.

4.4 Calibrage de fonctions de sensibilité par la technique epsilon-MOPSO

Dans cette section, on propose une deuxième formulation pour la synthèse des correcteurs digitaux RST. La technique epsilon-MOPSO sera utilisée comme outil principal de résolution du problème d'optimisation formulé. La stratégie proposée consiste à définir et calculer une fonction de sensibilité de sortie désirée.

4.4.1 Formulation du problème d'optimisation

Le problème de synthèse consiste à calculer un contrôleur digital RST qui garantit la stabilité interne de la boucle fermée et satisfait certaines contraintes fréquentielles sur la fonction de sensibilité de sortie.

Pour résoudre ce problème, d'abord la fonction de la sensibilité désirée est calculée en satisfaisant des contraintes de robustesse liées au nome $\mathcal{H}_\infty$ du filtre de pondération y est relatif. L'utilisation d'une partie fixe appropriée de la fonction de sensibilité de sortie désirée assurera un placement des pôles auxiliaires du régulateur RST. L'inverse cette fonction de sensibilité désirée définie alors son filtre de pondération $\mathcal{H}_\infty$.

4.4.2 Fonctions de sensibilité désirées

Comme on a montré dans la section 4.2, la fonction de sensibilité de sortie d'un procédé muni d'un correcteur polynomial RST est définie sous la forme suivante :

$$S_{yp}\left(z^{-1}\right) = \frac{A\left(z^{-1}\right)S\left(z^{-1}\right)}{A\left(z^{-1}\right)S\left(z^{-1}\right) + z^{-d}B\left(z^{-1}\right)R\left(z^{-1}\right)} = \frac{A\left(z^{-1}\right)S\left(z^{-1}\right)}{P_D\left(z^{-1}\right)P_F\left(z^{-1}\right)} \tag{4.39}$$

avec P_D et P_F représentent respectivement deux polynômes issus de la décomposition en pôles dominants et pôles auxiliaires de la boucle fermée. Comme les racines des polynômes A et P_D sont connus (pôles de la boucle fermée), nous pouvons les fixer dans la fonction de sensibilité dite désirée. Cette fonction de transfert peut être alors réécrite comme suit :

$$S_d\left(z^{-1}\right) = \frac{A\left(z^{-1}\right)}{P_D\left(z^{-1}\right)} \cdot \frac{X\left(z^{-1}\right)}{Y\left(z^{-1}\right)} \tag{4.40}$$

Les termes X et Y sont deux polynômes inconnus qui présentent les variables de décision du problème d'optimisation considéré. Ces variables, données dans l'équation (4.39) seront déterminer en utilisant les algorithmes d'optimisation multi-objectifs développés :

$$\begin{cases} X\left(z^{-1}\right) = \left(1 - x_1 z^{-1}\right)\left(1 - x_2 z^{-1}\right) \cdots \left(1 - x_{nx} z^{-1}\right) \\ Y\left(z^{-1}\right) = \left(1 - y_1 z^{-1}\right)\left(1 - y_2 z^{-1}\right) \cdots \left(1 - y_{ny} z^{-1}\right) \end{cases} \tag{4.41}$$

Il est démontré dans [Kwakernaak, 1986] que le dénominateur de la sensibilité S_d apparaîtra comme une partie des derniers pôles en boucle fermée, si l'inverse S_d est utilisé comme un filtre de pondération $\mathcal{H}_\infty$. Cette technique, connue sous le nom du placement de pôles partiels justifie l'interprétation du filtre de pondération comme l'inverse de la fonction de sensibilité souhaitée [Kwakernaak, 1986 ; Postlethwaite et al., 1990]. D'autre part, le choix naturel du polynôme A dans le numérateur de la fonction de transfert S_d empêchera la présence des pôles en boucle ouverte en tant que pôles en boucle fermée.

Le vecteur de décision du problème d'optimisation est noté $\boldsymbol{\theta} = [x_1,\ x_2, ...,\ x_{nx},\ y_1,\ y_2, ... , y_{ny}]$, dont les composantes x_1 à x_{nx} et y_1 à y_{ny} représentent respectivement les racines réelles des polynômes X et Y. La fonction de sensibilité, notée $S_d\left(z^{-1}, \boldsymbol{\theta}\right)$ vue qu'elle est fonction de ces variables de décision, est supposée propre.

Dans cette formulation, les parties fixes souhaitées dans le dénominateur du régulateur peuvent être mises directement dans le numérateur de S_d. En effet, pour rejeter les perturbations constantes dans à la sortie, un terme intégrateur doit être incorporé dans l'expression de la sensibilité $S_d\left(z^{-1}, \boldsymbol{\theta}\right)$. Ceci peut être facilement fait en choisissant $x_1 = 1$. L'ordre minimal des polynômes X et Y dépend de l'ordre des polynômes A et P_D ainsi que les parties fixes de Y, d'où on aura la condition suivante :

$$n_X \geq n_A + \left(n_Y\right)_{\min} - n_{P_D} \tag{4.42}$$

avec n_A, n_x, n_{P_D} et $\left(n_Y\right)_{min}$ sont respectivement les degrés des polynômes A, X , P_D et des parties fixes du polynôme Y. Si on choisit $n_{P_D} = n_A$ et avec un terme intégrateur dans le polynôme Y, on obtient $n_X = n_Y = 1$. Dans ce cas, il suffit de calculer un seul paramètre qui est le coefficient x_1 .

4.4.3 Critères de performance

Les pôles et les zéros instables de la boucle ouverte d'un système jouent un rôle très important pour la boucle fermée et par conséquent sur la forme de la fonction de sensibilité. Ainsi, ces derniers sont considérés pour la détermination de la fonction de sensibilité désirée. Le résultat qui explique la relation entre les pôles et les zéros instables de la boucle ouverte et la fonction de sensibilité est l'égalité de Zames-Francis basée sur la formule intégrale de Poisson issue de la théorie des fonctions complexes [Zames & Francis, 1983 ; Francis & Zames, 1984]. Le résultat qui suit est la version en temps discret de cette égalité, obtenue par Sung et Hara dans [Sung & Hara, 1988].

On suppose que $\beta = re^{j\phi}$ est un zéro instable de la boucle ouverte. Si le système en boucle fermée est stable, alors la relation suivante est vérifiée :

$$\pi \log \left| B_\alpha^{-1}(\beta) \right| = \int_{-\pi}^{\pi} \log \left| S_{yp}(j\omega) \right| \cdot \frac{r-1}{1-2r\cos(\omega-\phi)+r} \cdot d\omega \tag{4.43}$$

Dans cette expression, le terme $B_\alpha(\beta)$ est le produit de Blaschke qui est formé des pôles instables de la boucle ouverte :

$$B_\alpha(z) = \prod_i \frac{z-\alpha_i}{1-\overline{\alpha}_i z} \tag{4.44}$$

Puisque cette égalité est satisfaite pour S_{yp} quelconque, elle devrait être également pour la fonction de sensibilité désirée $S_d(z^{-1}, \boldsymbol{\theta})$ précédemment définie. Pour cela, on choisit cette égalité comme critère à optimiser dont le but d'avoir tout les paramètres inconnus des polynômes X et Y. La fonction coût à minimiser dans le problème multi-objectif traité peut être ainsi exprimée en fonction des variables de décision $\boldsymbol{\theta}$ comme suit :

$$F_i(\boldsymbol{\theta}) = \left| \int_{-\pi}^{\pi} \log \left| S_d(j\omega) \right| \cdot \frac{r_i^2-1}{1-2r_i^2\cos(\omega-\phi)+r_i^2} \cdot d\omega - \pi \log \left| B_\alpha^{-1}(\beta_i) \right| \right| \tag{4.45}$$

avec $\beta_i = r_i e^{j\phi_i}$ sont des zéros instables de la boucle ouverte.

En effet, seules les zéros instables du processus sont connus de sorte que le nombre des critères est égal au nombre des zéros instables. Le temps de retard dans les systèmes à temps discret est en effet un zéro instable à l'infini, ce qui signifie que r tend vers l'infini dans l'équation (4.42) et le critère d'optimisation devient :

$$F_d(\boldsymbol{\theta}) = \left| \int_{-\pi}^{\pi} \log \left| S_d(j\omega) \right| d\omega - \pi \sum \log \left| \alpha_i \right| \right| \tag{4.46}$$

Le problème de synthèse de correcteurs RST ainsi reformulé consiste alors à minimiser la fonction objectif multicritères suivante sous des contraintes définies ci-après dans le domaine fréquentiel :

$$F(\boldsymbol{\theta}) = [f_1(\boldsymbol{\theta}), f_2(\boldsymbol{\theta}), \ldots, f_{n_{uz}}(\boldsymbol{\theta}), f_d(\boldsymbol{\theta})] \tag{4.47}$$

où n_{uz} est le nombre des zéros instables du processus.

4.4.4 Contraintes d'optimisation

Dans le formalisme de la commande numérique RST, les marges de robustesse, de module et de retard, ainsi que les performances en termes de rejet de perturbation du système corrigé peuvent être définies dans le domaine fréquentiel sous forme des gabarits sur la fonction de sensibilité de sortie [Landau & Karimi, 1998 ; Zito, 2005]. Ces types des gabarits peuvent être traduits analytiquement sous formes de contraintes non linéaires pour le problème d'optimisation multi-objectif RST ainsi formulé. Ces contraintes sur le module de la fonction de sensibilité désirée auront alors la forme suivante :

$$C_L(\omega) \leq |S_d(j\omega, \boldsymbol{\theta})| \leq C_U(\omega) \tag{4.48}$$

avec $C_L(\omega)$ et $C_U(\omega)$ représentent respectivement les bornes inférieures et supérieures sur la réponse fréquentielle de la sensibilité désirée en boucle fermée.

4.4.5 Problème de synthèse RST multi-objectif formulé

Le programme d'optimisation multi-objectif, à résoudre moyennant les algorithmes métaheuristiques considérés pour la synthèse de correcteurs polynomiaux RST, est finalement défini comme suit :

Etape 1 : définition de la structure de la fonction de sensibilité désirée ainsi que les variables de décision à chercher :

$$S_d\left(z^{-1}\right) = \frac{A\left(z^{-1}\right)}{P_D\left(z^{-1}\right)} \cdot \frac{\left(1 - x_1 z^{-1}\right)\left(1 - x_2 z^{-1}\right) \cdots \left(1 - x_{nx} z^{-1}\right)}{\left(1 - y_1 z^{-1}\right)\left(1 - y_2 z^{-1}\right) \cdots \left(1 - y_{ny} z^{-1}\right)} \tag{4.49}$$

$$\boldsymbol{\theta} = [x_1,\ x_2, \ldots,\ x_{nx},\ y_1,\ y_2, \ldots, y_{ny}] \in \Re^{(nx+ny)} \tag{4.50}$$

Etape 2 : définition du coût multi-objectif à minimiser via l'indice de performance retenu en fonction de l'égalité de Zames-Francis, Equation (4.45).

Etape 3 : définition analytique des expressions des gabarits de robustesse sur la réponse fréquentielle de la fonction de sensibilité désirée. La forme générique du problème d'optimisation RST multi-objectif est alors la suivante :

$$\underset{\substack{\boldsymbol{\theta}=[x_1,x_2,\ldots,x_{nx},\,y_1,y_2,\ldots,y_{ny}]^T\in\Re^{(nx+ny)}\\ \text{sous contraintes:}}}{\text{minimiser}} \quad \begin{array}{c}((f_1(\boldsymbol{\theta}),f_2(\boldsymbol{\theta}),\ldots,f_{n_{uz}}(\boldsymbol{\theta})),f_d(\boldsymbol{\theta}))\\ g_1(\theta)=|S_{yp}(z^{-1},\boldsymbol{\theta})|-|W^{-1}|_{\sup}\leq 0\\ g_2(\boldsymbol{\theta})=|W^{-1}|_{\inf}-|S_{yp}(z^{-1},\boldsymbol{\theta})|\leq 0\end{array} \tag{4.51}$$

Etape 4 : inverser l'expression obtenue pour la fonction de sensibilité désirée $S_d(z^{-1})$ et définir le filtre de pondération $\mathcal{H}_\infty$ pour la synthèse RST.

4.4.6 Mise en oeuvre de l'approche de synthèse proposée

4.4.6.1 Description du système à commander

Le système de transmission flexible, pris ici comme un banc de test pour la stratégie de commande epsilon-MOPSO RST proposée, est un processus comportant trois poulies horizontales reliées par deux courroies élastiques, figure 4.10. La première poulie du système est entraînée par un moteur à courant continu dont la position est commandée par contre-réaction. La troisième poulie peut être chargée par des petits disques de différents poids. L'entrée du système est la référence pour la position de l'axe de la première poulie. La sortie étant la position de l'axe de la troisième poulie mesurée moyennant un capteur de position. Un ordinateur est utilisé pour la commande numérique RST du système étudié [Landau & Karimi, 1998].

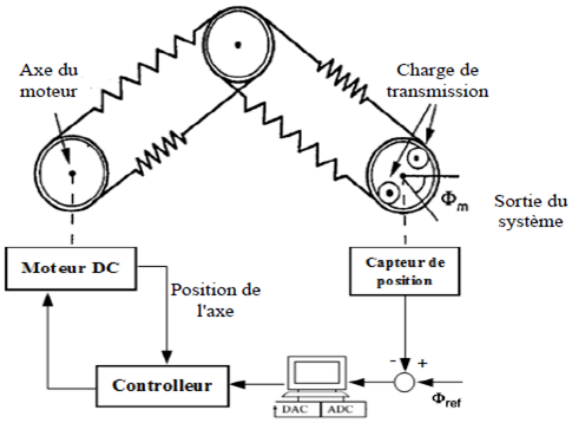

FIGURE 4.10 – Diagramme schématique de la transmission flexible [Landau & Karimi, 1998].

L'intérêt porté à ce système réside dans le faite qu'il est largement utilisé dans la littérature pour la commande RST classique, donc il permet de servir comme base de comparaison des résultats obtenus, ainsi que les faibles modes de vibration amorties et les grandes variations de leurs fréquences avec la charge que présente un tel procédé. Les modèles à temps discret du processus de transmission flexible, obtenus avec une période d'échantillonnage égale à $T_s = 50ms$, pour deux situations extrêmes (sans charge et à pleine charge), ainsi que pour une situation intermédiaire, sont donnés par les équations suivantes :

– Modèle sans charge (0 %) :

$$\begin{aligned} A_0\left(z^{-1}\right) &= 1 - 1.41833z^{-1} + 1.58939z^{-2} - 1.31608z^{-3} + 0.88642z^{-4} \\ B_0\left(z^{-1}\right) &= 0.28261z^{-1} + 0.50666z^{-2}; \quad d = 2 \end{aligned} \tag{4.52}$$

– Modèle à demi-charge (50%) :

$$\begin{aligned} A_1\left(z^{-1}\right) &= 1 - 1.99185z^{-1} + 2.20265z^{-2} - 1.84083z^{-3} + 0.89413z^{-4} \\ B_1\left(z^{-1}\right) &= 0.10276z^{-1} + 0.18123z^{-2}; \quad d = 2 \end{aligned} \tag{4.53}$$

– Modèle à pleine charge (100 %) :

$$\begin{aligned} A_2\left(z^{-1}\right) &= 1 - 2.0967z^{-1} + 2.3196z^{-2} - 1.93353z^{-3} + 0.87129z^{-4} \\ B_2\left(z^{-1}\right) &= 0.06408z^{-1} + 0.10407z^{-2}; \quad d = 2 \end{aligned} \tag{4.54}$$

4.4.6.2 Résultats de simulation et discussion

Dans cette partie, on présente toute les étapes de synthèse du régulateur digital RST reformulée sous forme d'un problème d'optimisation multi-objectif. La résolution d'un tel problème est achevée moyennant les algorithmes NSGA, MODE, MOPSO proposés et plus particulièrement, l'algorithme perfectionné epsilon-MOPSO. La figure 4.11 montre les réponses fréquentielles du système de transmission flexible étudié en boucle ouverte. Ces courbes de Bode, de phase et de module, montrent les deux modes de vibration des transmissions du système ainsi que leurs variations en fonction de la charge raccordée. Le modèle sans charge est considéré comme le modèle nominal. Pour tous les modèles retenus du système de transmission flexible, la fonction de sensibilité désirée à déterminer prendra la forme suivante :

$$S_d\left(z^{-1}\right) = \frac{A\left(z^{-1}\right)}{P_D\left(z^{-1}\right)} \cdot \frac{\left(1 - x_1 z^{-1}\right)\left(1 - x_2 z^{-1}\right)}{\left(1 - y_1 z^{-1}\right)\left(1 - y_2 z^{-1}\right)} \tag{4.55}$$

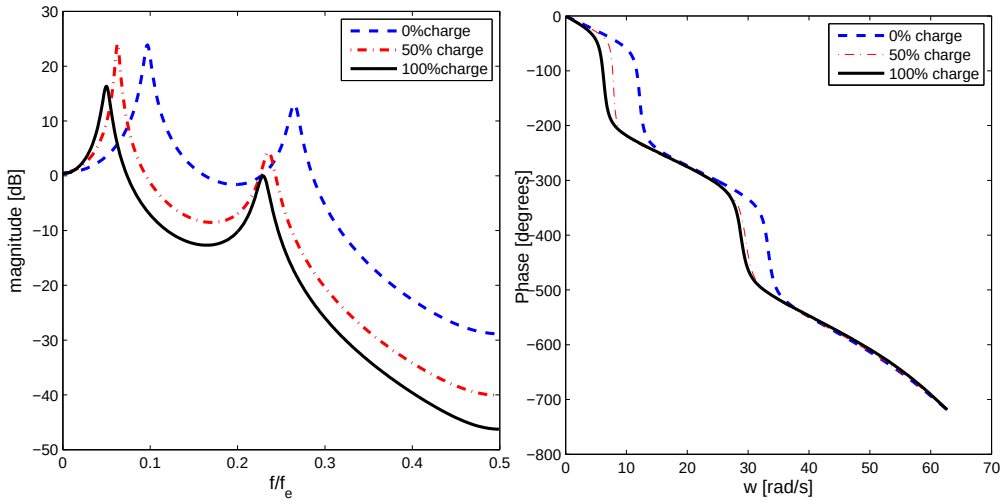

FIGURE 4.11 – Les réponses fréquentielles des modèles en boucle ouverte.

On note que l'utilisation d'une partie fixe appropriée de la fonction de sensibilité de sortie désirée assurera un placement des pôles auxiliaires de la structure de commande RST. Il est donc possible d'obtenir les autres pôles en boucle fermée à proximité du cercle unité. Pour éviter ce problème, on peut changer le domaine de stabilité en choisissant un cercle ayant un rayon inférieur. Le polynôme P_D est défini pour tous les modèles du système par l'équation suivante :

$$P_D\left(z^{-1}\right) = \left(1 - 1.1396z^{-1} + 0.3734z^{-2}\right)\left(1 - 0.0886z^{-1} + 0.5135z^{-2}\right) \tag{4.56}$$

Afin d'obtenir d'introduire un terme intégrateur dans le régulateur RST, la valeur de la variable de décision x_1 sera fixée à 1. Cela signifie que le filtre de pondération aura un pôle dans le cercle unité, qui n'est pas autorisé dans l'optimisation de la norme $\mathcal{H}_\infty$. Pour éviter ce problème, on peut choisir $x_1 = 0.9999$. Il reste alors à calculer trois paramètres dans l'expression de la sensibilité désirée par l'algorithme epsilon-MOPSO. Par conséquent, le vecteur de décision sera défini comme suit :

$$\boldsymbol{\theta} = [x_2,\, y_1,\, y_2]^T \in \Re^3 \tag{4.57}$$

Cette étude est basée essentiellement sur la détermination des pôles et surtout des zéros des modèles pour pouvoir définir la taille du problème à optimiser. D'après la figure 4.12, on peut constater qu'on est face à un problème d'optimisation bi-objectif. D'après ces figures ainsi que le tableau 4.4, chaque modèle dispose d'un seul zéros instable, d'où

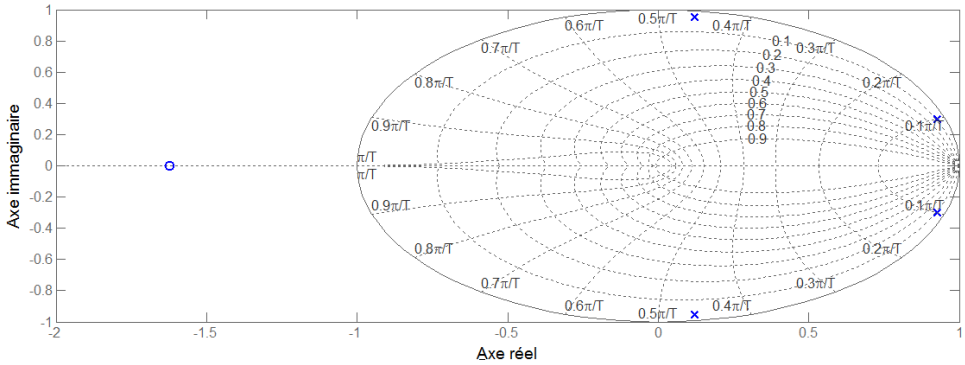

FIGURE 4.12 – Emplacement des pôles et des zéros du modèle sans charge.

l'indice de performance à minimiser dans le problème d'optimisation multi-objectif pour la synthèse RST est défini comme suit :

$$F(\boldsymbol{\theta}) = \left[\; f_1(\boldsymbol{\theta}), \quad f_d(\boldsymbol{\theta}) \; \right] \tag{4.58}$$

Dans le formalisme d'optimisation avec métaheuristiques et vu l'aspect stochastique

TABLE 4.4 – Pôles et zéros du système de transmission flexible.

Charge	Pôles	Zéros
$0\%charge$	$-0.0917 \pm 0.9604i$ $0.8009 \pm 0.5577i$	-1.7928
$50\%charge$	$0.0853 \pm 0.9552i$ $0.9106 \pm 0.3781i$	-1.7636
$100\%charge$	$0.1221 \pm 0.9507i$ $0.9262 \pm 0.3008i$	-1.6241

et non reproductible des algorithmes utilisés, il est toujours nécessaire d'exécuter plusieurs fois le programme d'optimisation. Ceci est toutefois indispensable afin de pouvoir récupérer des résultats statistiques pouvant conclure sur la qualité des solutions obtenues et donc valider l'approche proposée. Dans cette étude de cas, nous choisissons d'exécuter 10 fois l'algorithme epsilon-MOPSO proposé avec une valeur de ε fixée à 0.015.

Comme montré dans la figure 4.13, le temps de calcul de l'algorithme epsilon-MOPSO pour les 10 tests effectués sur le premier modèle du système (0% charge) varie entre 100 et 120 secondes. Nous signalons ici que l'algorithme NSGA-II utilisé pour la résolution

du même problème présente un temps de calcul relativement plus élevé que les autres algorithmes (MODE, MOPSO et epsilon-MOPSO).

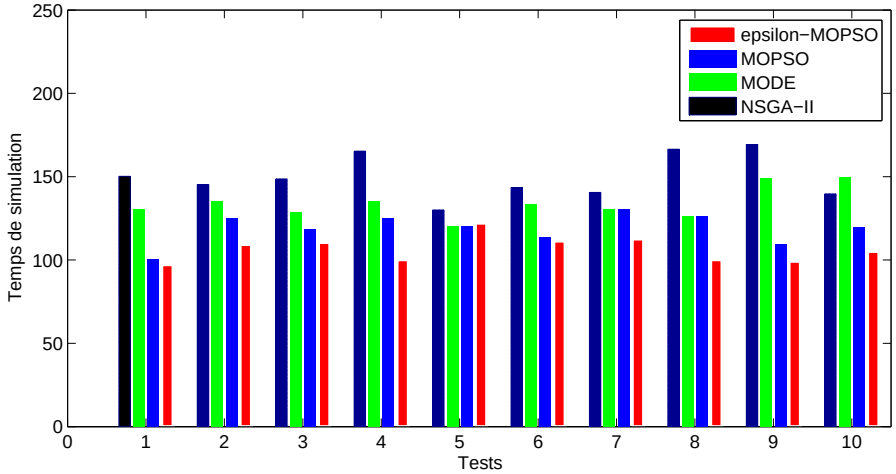

FIGURE 4.13 – Temps de calcul des algorithmes pour l'optimisation du correcteur RST : cas du premier modèle du système.

TABLE 4.5 – Coefficients optimisés de la fonction de sensibilité désirée RST.

Algorithme	Modèle du système	x_2^*	y_1^*	y_2^*
Epsilon-MOPSO	0 % charge	-0.3743	0.9224	0.4414
MOPSO	0 % charge	-0.34981	0.9703	0.42991
NSGA-II	0 % charge	-0.2978	0.9229	0.3751
MODE	0 % charge	-0.4077	0.8847	0.5049

La convergence des métaheuristiques utilisées pour la résolution du problème d'optimisation RST, en particulier l'algorithme perfectionné epsilon-MOPSO, a toujours lieu dans la même région de l'espace de recherche quelle que soit la population de départ. Ceci indique que les algorithmes implémentés réussissent, bien évidement avec des performances en convergence comparables, à trouver souvent la région la plus intéressante à exploiter de l'espace de recherche. La surface de compromis obtenue dans cette application est donnée dans la figures 4.14. Le domaine de stabilité est réduite à un cercle de rayon 0.85 centré à $\alpha = 0.15$. Ensuite, les équations des modèles et les filtres de pondération sont transformés en conséquence à ce nouveau domaine de stabilité. On note aussi que le domaine de stabilité réduit doit contenir le point $[1,\ 0j]$ afin de maintenir l'intégrateur dans le filtre

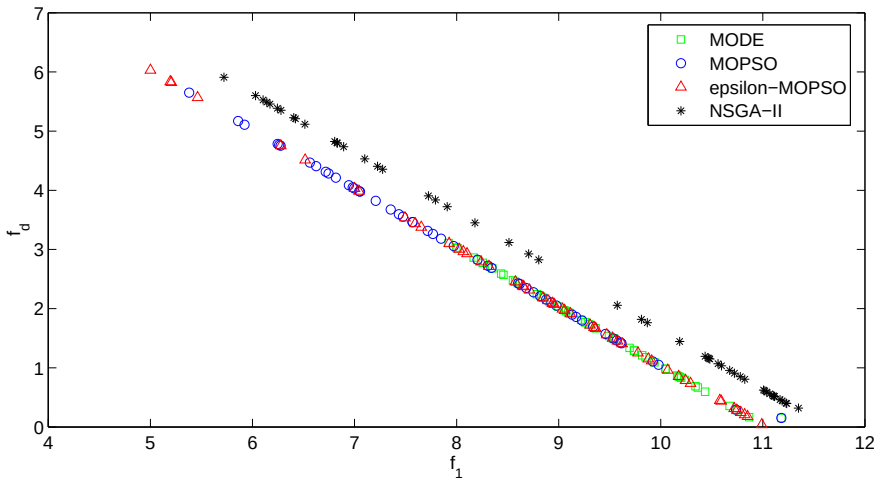

FIGURE 4.14 – Fronts de Pareto des algorithmes pour le premier modèle du système.

de pondération. Les inverses des fonctions de sensibilités désirées S_d^{-1}, formant le filtre de pondération, est donné par l'équation (4.59) et obtenu par l'approche d'optimisation epsilon-MOPSO :

– Modèle à 0 % de charge :

$$S_d^{-1}(z^{-1}) = \frac{P_D(1-0.9703z^{-1})(1-0.4414z^{-1})}{A_0(1-z^{-1})(1+0.3743z^{-1})} \tag{4.59}$$

La norme $\mathcal{H}_\infty$ des filtres de pondération obtenus $S_d^{-1}(z^{-1})$ $S_{yp}^{-1}(z^1)$ doit être minimisée. En utilisant la technique de la paramétrisation de Youla-Kucera, nommée la Q-paramétrisation, la fonction de transfert S_{yp}^{-1} est définie comme suit [Youla et al. 1976 ; Maciejowski, 1989] :

$$S_{yp}^{-1}(z^1) = 1 - \frac{B(z^{-1})}{A(z^{-1})}Q(z^{-1}) \tag{4.60}$$

Pour chercher les coefficients du polynôme $Q(z^{-1})$, le recours à l'optimisation par l'algorithme PSO présente la solution proposée [Madiouni et al., 2013 ; Bouallègue et al. 2012].

Donc, il reste à minimiser le critère d'optimisation défini par l'équation suivante :

$$\underset{\boldsymbol{\theta}=[q_0,q_1,q_2]^T\in\Re^3}{\text{minimiser}} \quad \left(\left\| S_d^{-1}(z^{-1}) - \frac{S_d^{-1}(z^{-1})B(z^{-1})}{A(z^{-1})}Q(z^{-1}) \right\|_\infty\right) \tag{4.61}$$

Après, les paramètres du régulateur RST dans le domaine de stabilité réduit sont définis

par l'expression suivante :

$$\frac{R(z^{-1})}{S(z^{-1})} = \frac{Q(z^{-1})}{1 - Q(z^{-1})\frac{B_0(z^{-1})}{A_0(z^{-1})}} \tag{4.62}$$

Finalement, nous obtenons les paramètres du régulateur digital RST pour les trois modèles du système de transmission flexible étudié :

$$\begin{cases} R(z^{-1}) = 0.4563 + 1.2124z^{-1} + 0.8992z^{-2} \\ S(z^{-1}) = 1 - 0.7231z^{-1} + 0.4211z^{-2} \end{cases} \tag{4.63}$$

Les réponses fréquentielles des fonctions de sensibilité de sortie et d'entrée obtenues sont montrées respectivement dans figures 4.15 et 4.16. Nous remarquons que les contraintes du problème d'optimisation, traduits graphiquement sous forme des gabarits fréquentiels sur les lieux des Bode, sont vérifiées pour les différents modèles du processus commandé.

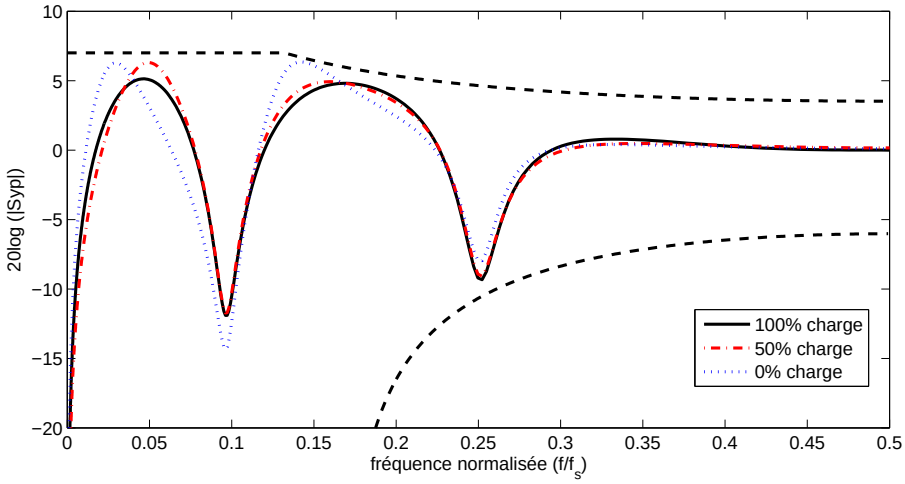

FIGURE 4.15 – Réponse fréquentielle de la fonction de sensibilité de sortie du système.

Pour observer et analyser les performances du système corrigé dans le domaine temporel, nous présentons dans la figure 4.17 les réponses indicielles de ce dernier avec les correcteurs polynomiaux RST calculés. Grossièrement, les performances du système corrigé en termes de régulation et de poursuite sont bien satisfaisantes. Le rejet de perturbations externes sur la position de l'axe de la troisième poulie, sortie asservie du système, est garantie avec une dynamique plus ou moins lente en relation avec le type de la charge.

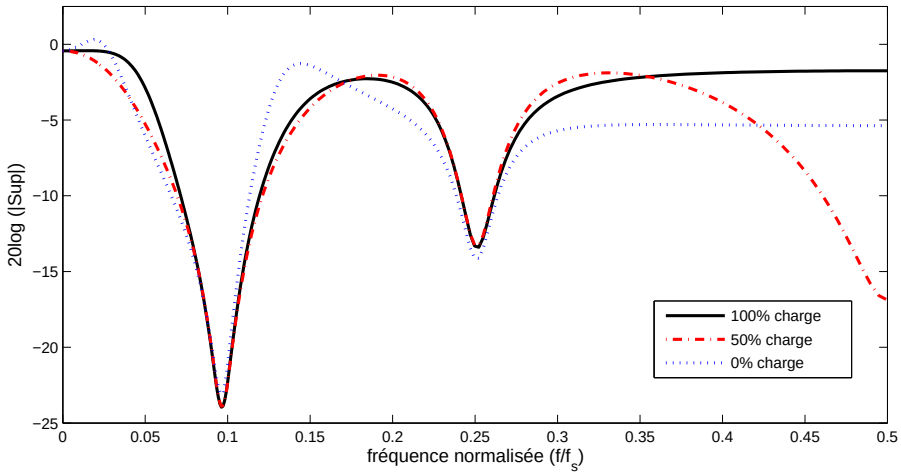

FIGURE 4.16 – Réponse fréquentielle de la fonction de sensibilité d'entrée du système.

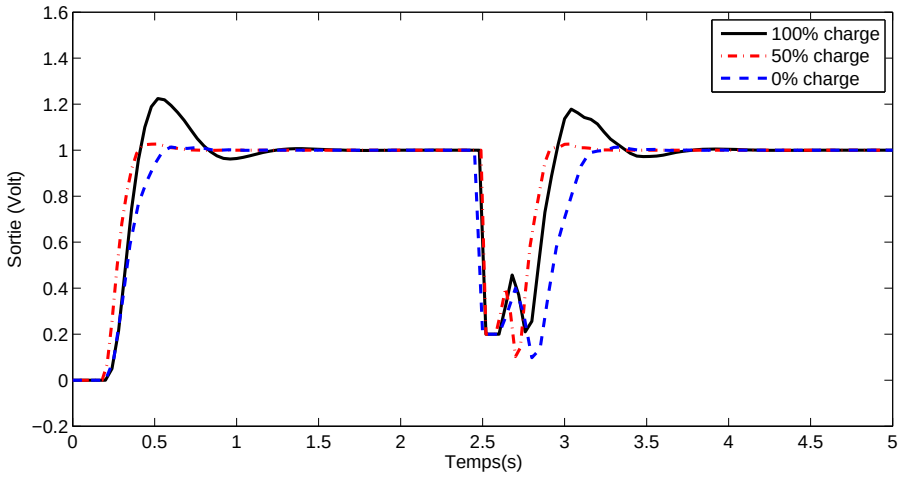

FIGURE 4.17 – Réponses indicielles du système avec le correcteur RST optimisé.

4.5 Conclusion

Deux nouvelles approches de synthèse de correcteurs robustes à structure RST, exploitant la technique epsilon-MOPSO, sont proposées et appliquées avec succès dans ce chapitre. La première formulation de synthèse RST consiste à minimiser des critères de performance dans le domaine temporel issus de la théorie de la commande optimale. Ces critères d'optimalité, de type IAE et MO, sont minimisés sous des contraintes non analytiques liées à la réponse indicielle du système corrigé en termes de dépassement, de temps de réponse et d'erreurs statique en régime établie. Cette solution est appliquée avec succès pour la commande RST en vitesse d'un moteur DC. La deuxième méthode proposée est basée une formulation du problème de réglage RST dans le plan fréquentiel moyennant l'exploitation du concept d'égalités de Zames-Francis. La minimisation des coûts de robustesse, en forme intégrale de Poisson, a permis la détermination d'une fonction de sensibilité de sortie désirée. L'inverse d'une telle fonction de transfert est retenu comme filtre de pondération pour la minimisation de la norme $\mathcal{H}_\infty$ du modèle en boucle fermée. La mise en oeuvre par simulation numérique sur un système de transmission flexible à charges variables a montré la validité et l'efficacité de la stratégie de commande élaborée.

Conclusion générale

Les travaux de recherche consignés dans ce mémoire de thèse portent sur la synthèse et l'optimisation multi-objectif de correcteurs numériques à structure polynomiale RST par approches relevant de l'optimisation par métaheuristiques.

La proposition des nouvelles approches hybrides de synthèse, systématique et à complexité réduite, de lois de commande numérique à structure canonique RST constitue la principale contribution de ces travaux. Les problèmes de synthèse considérés sont formulés sous forme des problèmes d'optimisation multi-objectif sous contraintes de robustesse et de performances nominales auxquels des algorithmes métaheuristiques, essentiellement d'optimisation par essaims particulaires MOPSO et epsilon-MOPSO, ont été adaptés et perfectionnés. Les problèmes classiques de placement des pôles de la boucle fermée et de calibrage itératif des fonctions de sensibilités de correcteurs RST, le plus souvent difficiles et lourds en termes de temps de conception et de ressources de calcul, sont à présent résolus moyennant une systématisation des procédures de synthèse à base de la théorie d'optimisation difficile par métaheuristiques. Deux formulations du problème de synthèse RST, l'une par minimisation des critères de performance temporels liés à la réponse indicielle du système corrigé et issus de la théorie de la commande optimale et l'autre par optimisation d'une fonction de sensibilité désirée en satisfaisant des contraintes de robustesse $\mathcal{H}_\infty$, ont été proposées.

Le premier chapitre du mémoire a été consacré à la présentation des généralités sur la théorie de la commande robuste ainsi que les principales techniques de synthèse relevant de ce formalisme. Des études ont été menées dans le but de montrer la complexité liée à la résolution des tels problèmes de synthèse par des méthodes classiques connues. Notre intérêt s'est porté plus particulièrement à la commande numérique robuste à structure canonique RST de systèmes complexes. Nous avons clôturé ce chapitre par la présentation des notions de bases liées à la théorie de l'optimisation difficile par métaheuristiques. Une classification de ces techniques approchées d'optimisation, de voisinage et d'autres à base de population, a été présentée. Enfin, le recours à l'optimisation multi-objectif, comme

solution proposée pour la problématique abordée des travaux, a été justifiée.

Nous avons présenté dans le deuxième chapitre quelques définitions de base et des notations formelles sur les algorithmes évolutionnaires, plus particulièrement le NSGA-II et le MODE, en optimisation multi-objectif. Les notions liées à la modélisation d'un problème d'optimisation multi-objectif sous contraintes, d'une part, et à la dominance au sens de Pareto des solutions en optimisation multicritères ont été présentées, d'autre part. Ce chapitre a présenté aussi une analyse des algorithmes génétiques NSGA, et puis NSGA-II, ainsi que les algorithmes à évolution différentielle MODE dans le but d'être adaptés aux problèmes de synthèse RST reformulés en quatrième chapitre. Quelques métriques de mesure de performances des métaheuristiques en multi-objectif ont été définies afin de pouvoir tirer des conclusions sur la validité des algorithmes proposés.

Le troisième chapitre a été dédié à l'étude et au perfectionnement d'un algorithme MOPSO d'optimisation par essaims particulaires multi-objectif qui a servi d'outil de base pour la résolution des problèmes de synthèse RST reformulés. Le concept d'epsilon-dominance est ensuite introduit en vue d'améliorer les capacités de l'algorithme epsilon-MOPSO obtenu en termes de rapidité de convergence et de diversité des solutions fournies sur les surfaces de Pareto. Une mise en œuvre des algorithmes MOPSO, epsilon-MOPSO, NSGA-II et MODE ainsi proposés et perfectionnés pour l'optimisation de certaines fonctions de test issues de la littérature a été effectuée. Nous avons mené également des comparaisons des résultats obtenus par ces différents algorithmes ainsi que du calcul des métriques de performance définies.

Le quatrième et dernier chapitre a présenté une application des métaheuristiques proposées, en particulier l'algorithme perfectionné epsilon-MOPSO, pour la synthèse systématique et l'optimisation de correcteurs numériques à structure polynomiale RST. Deux nouvelles formulations multicritères de la synthèse RST ont été proposées afin de surmonter les problèmes délicats et souvent difficiles de placement des pôles de la boucle fermée et de calibrage des fonctions de sensibilité. Une première formulation, traduite dans le domaine temporel, est basée sur la minimisation des certains indices de performance issus de la théorie de la commande optimale comme les critères IAE (*Integral Absolute Error*) et MO (*Maximum Overshoot*) sous des contraintes non analytiques en les paramètres et liés essentiellement à la réponse indicielle du système corrigé. Dans la deuxième approche, basée sur les concepts de l'égalité de Zames-Francis, nous avons proposé une formulation dans le domaine fréquentiel qui consiste à définir et calculer les fonctions de sensibilité désirées sous contraintes de robustesse de type $\mathcal{H}_\infty$. Des applications, par simulations

numériques, sur un moteur électrique de type DC et un procédé de transmission flexible à charges variables, sont effectuées dans le but de montrer l'efficacité et la supériorité de la stratégie de commande hybride RST proposée.

Les résultats obtenus dans nos travaux de recherche sont encourageants et prometteurs dans le cadre de la systématisation des procédures de synthèse et l'hybridation des lois de commande robuste RST par des approches métaheuristiques globales et multi-objectif de type MOPSO. Néanmoins, il s'avère intéressant de développer davantage, en tant que perspectives, certains autres aspects liés essentiellement à :

- l'extension des approches proposées aux systèmes complexes : non linéaires, multivariables, incertains, etc. ;
- la synthèse de correcteurs robuste RST avec contrainte sur l'ordre en vue de réduire la complexité de la commande ;
- la généralisation des algorithmes d'optimisation multi-objectif proposés à d'autres domaines de l'automatique, tels que l'identification, le diagnostic et la commande tolérante aux défauts ;
- l'amélioration des techniques d'optimisation multi-objectif MOPSO en introduisant d'autres mécanismes de perfectionnement tels que le pPSO (perturbed PSO), le CGPSO (PSO à convergence garantie), voire les approches multi-essaims.

Annexe : Fonctions de test pour l'optimisation MO

TABLE 1 : Fonctions de problèmes bi-objectif [Coello.Coello et al., 2007].

Notation	Expression mathématique	Paramètres
F1	$\begin{cases} f_1(x_1,x_2) = x_1 \\ f_2(x_1,x_2) = \frac{g(x_1,x_2)}{x_1} \end{cases}$ avec $0 \leq x_1 \leq 1,\ -30 \leq x_2 \leq 30$ $g(x_1,x_2) = 11 + x_2^2 - 10\cos(2\pi x_2)$ $h(x_1,x_2) = \begin{cases} 1 - \sqrt{\frac{f(x_1,x_2)}{g(x_1,x_2)}}, & f(x_1,x_2) \leq g(x_1,x_2) \\ 0, & sinon \end{cases}$	$nPop = 100$ nIter $= 100$
F2	$\begin{cases} f_1(x) = \sum_{i=1}^{n-1} \left(-10\exp\left(-20\sqrt{x_i^2 + x_{i+1}^2}\right)\right) \\ f_2(x) = \sum_{i=1}^{n} \left(\lvert x_i\rvert^{0.8} + 5\sin(x_i^3)\right) \end{cases}$ avec $-5 \leq x \in \Re^3 \leq 5$	$nPop = 100$ nIter $= 100$
F3	$\begin{cases} f_1(x_1,x_2) = x_1 \\ f_2(x_1,x_2) = \frac{g(x_1,x_2)}{x_1} \end{cases}$ avec $0.1 \leq x_1,\ x_2 \leq 1,$ $g(x_1,x_2) = 2 - \exp\left(\frac{x_2-0.2}{0.004}\right)^2 - 0.8\exp\left(\frac{x_2-0.6}{0.4}\right)^2$	$nPop = 100$ nIter $= 100$
F4	$\begin{cases} f_1(x_1,x_2) = -x_1^2 + x_2 \\ f_2(x_1,x_2) = 0.5x_1 + x_2 + 1 \end{cases}$ avec $\frac{1}{6}x_1 + x_2 - \frac{13}{2} \leq 0,\ \frac{1}{2}x_1 + x_2 - \frac{15}{2} \leq 0,\ x_1 + x_2 - 30 \leq 0$	$nPop = 100$ nIter $= 100$

TABLE 2 : Fonctions de problèmes tri-objectif [Coello.Coello et al., 2007 ; Mostaghim & Teich, 2003].

Notation	Expression mathématique	Paramètres
F5	$\begin{cases} f_1(x_1) = x_1 \\ f_2(x) = g(x_2, ..., x_n)\, h(f_1, g) + 1 \end{cases}$ avec $g(x_2, ..., x_n) = 1 + \frac{9}{(n-1)} \left(\sum_{i=2}^{n} x_i \right)$ $h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right) \sin(10\pi f_1)$	$x_i \in [\![0, 1]\!]$ $nPop = 100$ $nIter = 200$
F6	$\begin{cases} f_1(x) = (1 + x_2) \cos\left(\frac{x_1 \pi}{2}\right) \cos\left(\frac{x_2 \pi}{2}\right) \\ f_2(x) = (1 + x_2) \cos\left(\frac{x_1 \pi}{2}\right) \sin\left(\frac{x_2 \pi}{2}\right) \\ f_3(x) = (1 + x_2) \sin\left(\frac{x_1 \pi}{2}\right) \end{cases}$	$x_i \in [\![0, 1]\!]$ $nPop = 500$ $nIter = 10$
F7	$\begin{cases} f_1(x) = x_1 \\ f_2(x) = x_2 \\ f_3(x) = 3.5 - \sum_{i=1}^{n} 1^n 2 x_i \sin(n\pi x_i) \end{cases}$	$x_i \in [\![0, 1]\!]$ $nPop = 500$ $nIter = 10$

Bibliographie

Abbass, H. A. (2002). The self-adaptive pareto differential evolution. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 831–836, New Jersey.

Abbass, H. A., Sarker, R., and Newton, C. (2001). PDE : A Pareto-frontier differential evolution approach for multi-objective optimization problems. In *Proceedings of the Congress on Evolutionary Computation (CEC'2001)*, volume 2, pages 971–978, New Jersey.

Abdennour, R. B., Borne, P., Ksouri, M., and M'Sahli, F. (2001). *Identification et commande numérique des procédés industriels*. Technip, Paris.

Adeyemo, J. A. and Otieno, F. A. O. (2009). Multi-objective differential evolution for solving engineering problems. *Journal of Applied Sciences*, 9(20) :3652–3661.

Ali, M., Pant, M., and Abraham, A. (2009). A modified differential evolution algorithm and its application to engineering problems. In *Proceedings of the International Conference of Soft Computing and Pattern Recognition (SoCPaR-2009)*, volume 2, pages 196–201, Malacca.

Ali, M., Siarry, P., and Pant, M. (2012). An efficient Differential Evolution based algorithm for solving multi-objective optimization problems. *European Journal of Operational Research*, 217(2) :404–416.

Andersson, J. and Wallace, D. (2002). Pareto optimization using the struggle genetic crowding algorithm. *Engineering Optimization*, 34(6) :623–643.

Angeline, P. (1998). Evolutionary optimization versus particle swarm optimization : Philosophy and performance differences. In *Proceedings of the 7th International Conference on Evolutionary Programming VII*, pages 601–610.

Angelis, G. Z. (2001). *System Analysis : Modeling and Control with Polytopic Linear Models*. PhD thesis, Technische Universiteit Eindhoven, Pays Bas.

Arzelier, D., Peaucelle, D., and Ariza, R. (2002). Une méthode itéerative pour la synthèse mixte $\mathcal{H}_2/\mathcal{H}_\infty$ par retour de sortie statique. In *Conférence Internationale Francophone d'Automatique CIFA'2002*, Nantes.

Astrom, K. J. and Wittenmark, B. (1995). *Adaptive Control.* PearsonEducation, North Asia.

Avriel, M. (1976). *Nonlinear Programming : Analysis and Methods.* Prentice-Hall, Englewood Cliffs, New Jersey.

Ayadi, M. (2002). *Contributions à la commande des systèmes linéaires plats de dimension finie.* Thèse de Doctorat, Institut National Polytechnique de Toulouse, Toulouse.

Ayadi, M., Haggège, J., Bouallègue, S., and Benrejeb, M. (2008). A Digital Flatness-based Control System of a DC Drive. *Studies in Informatics and Control*, 17(2) :201–214.

Back, T. (1996). *Nonlinear Programming : Analysis and Methods.* Oxford University Press, New York.

Bader, J. and Zitzler, E. (2008). *HypE : An algorithm for fast hypervolumebased manyobjective optimization.* TIK Report 286, Computer Engineering and Networks Laboratory (TIK), ETH Zurich.

Bagchi, T. P. (2001). Pareto-Optimal Solutions for Multi-objective Production scheduling problems. In *the First International Conference on Evolutionary Multi-Criterion Optimization*, pages 458–471, Switzerland.

Baghli, L., Razik, H., and Rezzoug, A. (1997). Neuro-Fuzzy controller in a field oriented control for induction Motors. In *Proceedings European Conference on Power Electronics and Applications EPE'97*, volume 1, pages 96–101, Trondheim, Norway.

Bandyopadhyay, S. and Mukherjee, A. (2015). An Algorithm for Many-Objective Optimization With Reduced Objective Computations : A Study in Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 19(3) :400–413.

Bernussou, J. (1996). *Commande robuste : développements et applications.* Hermés Science Publications, Paris.

Bernussou, J. and Oustaloup, A. (2001). *Conception de commandes robustes.* Hermés Science Publications, Paris.

Berro, A. (2001). *Optimisation multiobjectif et stratégies d'évolution en environnement dynamique.* Thèse de Doctorat, Université des Sciences Sociales Toulouse 1, Toulouse.

Berry, A. and Vamplew, P. (2005). The combative accretion model-multiobjective optimisation without explicit pareto ranking. In *C. A. Coello Coello, A. Hernandez Aguirre, and E. Zitzler, (Editors), Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, volume 3410, pages 77–91, Guanajuato, México.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont.

Beyer, H. G. (2001). *The theory of evolution strategies.* Springer, first edition, Germany.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Presses, Germany.

Blickle, T. and Thiele, L. (1995). A comparison of selection schemes used in genetic algorithms. *Evolutionary Computation*, 4(11) :311–347.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence : from natural to artificial systems.* Oxford University Presses, New York, USA.

Bonnet, S. (2008). *Approches numériques pour la commande des systèmes dynamiques.* Thèse de Doctorat, Université de Technologie de Compiègne, Compiègne.

Bonomi, E. and Lutton, J. L. (1988). Le recuit simulé. *Pour la Science*, 129 :68–677.

Borne, P. and Benrejeb, M. (2010). Des algorithmes d'optimisation-la nature, source d'inspiration pour l'ingénieur. l'essor des métaheuristiques pour l'optimisation difficile baséees sur le comportement de la nature. *L'ingènieur*, 129 :12–15.

Borne, P., Benrejeb, M., and Haggége, J. (2007). *Les réseaux de neurones : Présentation et applications.* Technip, Paris.

Borne, P., Dauphin-Tanguy, G., Richard, J. P., Rotella, F., and Zambettakis, I. (1992a). *Modélisation et identification des processus.* Tome 1, Technip, Paris.

Borne, P., Dauphin-Tanguy, G., Richard, J. P., Rotella, F., and Zambettakis, I. (1992b). *Modélisation et identification des processus.* Tome 2, Technip, Paris.

Borne, P., Dauphin-Tanguy, G., Richard, J. P., Rotella, F., and Zambettakis, I. (1993a). *Analyse et régulation des processus industriels.* Tome 1 : Régulation continue, Technip, Paris.

Borne, P., Dauphin-Tanguy, G., Richard, J. P., Rotella, F., and Zambettakis, I. (1993b). *Analyse et régulation des processus industriels.* Tome 2 : Régulation continue, Technip, Paris.

Bouallègue, S., Haggège, J., Ayadi, M., and Benrejeb, M. (2012). PID type Fuzzy Logic Controller Tuning Based on Particle Swarm Optimization. *Engineering Applications of Artificial Intelligence*, 25 :484–493.

Boussaid, I., Chatterjee, A., Siarry, P., and Nacer, M. A. (2012). Biogeography based optimization for constrained optimization problems. *Computers and Operations Research*, 39(12) :3293–3304.

Buhler, H. (1988). *Conception des systèmes automatiques, complément du traité de l'électricité*. Presse Polytechniques Romandes, Lausanne.

Caron, J. P. and Hautier, J. P. (1995). *Modélisation et commande de la machine asynchrone*. Edition Technip, Paris.

Carvalho, A. B. and Pozo, A. (2012). Measuring the convergence and diversity of CDAS Multi-Objective Particle Swarm Optimization Algorithms : A study of manyobjective problems. *Neurocomputing*, 75(1) :43–51.

Cerny, V. (1985). Thermodynamical Approach to the Travelling Salesman Problem : An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications*, 45(1) :41–51.

Chan, F. T. S. and Tiwari, M. K. (2007). *Swarm Intelligence : Focus on Ant and Particle Swarm Optimization*. Chapter 21, In Tech Education and Publishing, Vienne.

Chatterjee, A. and Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in Particle Swarm Optimization. *Computers and Operations Research*, 33(3) :859–871.

Chiu, S. Y., Sun, T. Y., Hsieh, S. T., and Lin, C. W. (2007). Cross searching strategy for Multi-Objective Particle Swarm Optimization. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 3135–3141, New York.

Chouiter, D. R. (1997). *Conception et réalisation d'une commande robuste de machine asynchrone*. Thèse de Doctorat, Ecole Centrale de Lyon, Lyon.

Clarke, D. W., Mohtadi, C., and Tuffs, P. S. (1987). Generalized Predictive Control, part I : The Basic Algorithm. *Automatica*, 23(2) :137–148.

Clerc, M. and Kennedy, J. (2002). The particle swarm : explosion, stability, and convergence in multi-dimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1).

Clerc, M. and Siarry, P. (2009). Une méthode inspirée de comportements coopératifs observés dans la nature : l'optimisation par essaim particulaire. *Revue d'Electricité et d'Electronique REE*, pages 25–32.

Coello.Coello, C. A. (2001). A Short Tutorial on Evolutionary Multiobjective Optimization. In *the First International Conference on Evolutionary Multi-Criterion Optimization*, pages 21–40.

Coello.Coello, C. A. (2002). Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms : A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(12) :1245–1287.

Coello.Coello, C. A., Lamont, G. B., and Van-Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation Series, Springer.

Coello.Coello, C. A. and Lechuga, M. S. (2002). MOPSO : A proposal for Multiple Objective Particle Swarm Optimization. In *Proceedings of the Evolutionary Computation (CEC'02)*, volume 2, pages 1051–1056, Washington.

Coello.Coello, C. A., Pulido, G. T., and Lechuga, M. S. (2004). Handling Multiple Objectives with Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3) :256–279.

Collette, Y. (2002). *Contribution à l'évaluation et au perfectionnement des méthodes d'optimisation multiobjectif : Application à l'optimisation des plans de rechargement de combustible nucléaire*. Thèse de Doctorat, Université de Paris 12, Créteil.

Collette, Y. and Siarry, P. (2004). *Multiobjective Optimization : Principles and Case Studies, Decision Engineering*. Decision Engineering. Springer-Verlag, Berlin.

Colorni, A., Dorigo, M., and Maniezzo, V. (1991). Distributed optimization by ant colonies. In *Proceedings of the First European Conference on Artificial Life*, pages 134–142, Paris.

Creutz, M. (1983). Micro-canonical Monte Carlo simulation. *Physical Review Letters*, 50(19) :1411–1415.

D. W. Clarke, C. M. and Tuffs, P. S. (1987). Generalized Predictive Control, part I : The Basic Algorithm. *Automatica*, 23(2) :137–148.

Daafouz, J. (1997). *Robustesse en performance des systèmes linéaires incertains : Placement de pôles et coût garanti*. Thèse de Doctorat, Institut National des Sciences Appliquées de Toulouse, Toulouse.

Dantzig, G. (1963). *Linear programming and extensions.* Princeton University Press, New Jersey.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life.* John Murray.

Day, R., Kleeman, M. P., and Lamont, G. (2004). Multi-objective fast messy genetic algorithm solving deception problems. In *Proceedings of the Conference on Evolutionary Computation*, volume 2, pages 1502–1509, DOI : 10.1109/CEC.2004.1331074.

Deb, K. (1999a). Multi-Objective Genetic Algorithms : Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3) :205–230.

Deb, K. (1999b). *An overview of multi-objective evolutionary algorithms.* Journée J.E.T., Copie de transparents.

Deb, K. (2000). *Introduction to selection.* In : Evolutionary computation 1 : advanced algorithms and operators, Institute of Physics Publishing, Bristol and Philadelphia.

Deb, K. and Jain, H. (2013). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Non-dominated Sorting Approach, Part I : Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4) :577– 601.

Deb, K., Mohan, M., and Mishra, S. (2005). Evaluating the ε-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *European Journal of Operational Research*, 185(3) :501–525.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multi objective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2001). *Scalable Test Problems for Evolutionary Multi-Objective Optimization.* Technical Report-112, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zurich.

Deb, K. and Tiwari, S. (2008). Omni-optimizer : A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research,*, 185(3) :1062–1087.

Delmotte, F. (1997). *Analyse multi-modèle.* Thèse de Doctorat, Université des Sciences et Techniques de Lille, Lille.

Dixon, L. C. W. and Szego, G. P. (1975). *Towards Global Optimization*, volume 1. Editors, North-Holland, Amsterdam.

Dixon, L. C. W. and Szego, G. P. (1978). *Towards Global Optimization*, volume 2. Editors, North-Holland, Amsterdam.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991). Positive feedback as a search strategy. Technical report, pages 91-16, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system : Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, part B*, 26(1) :29–41.

Doyle, J. C., Glover, K., Khargonekar, P., and Francis, B. A. (1989). State-Space Solutions to Standard $\mathcal{H}_2/$ and $\mathcal{H}_\infty$ Control Problems. *Evolutionary Computation*, 7(3) :205–230.

Dréo, J., Petrowski, A., Siarry, P., and Taillard, E. (2003). *Métaheuristiques pour l'optimisation Difficile*. Eyrolles, Paris.

Duc, G. and Font, S. (1999). *Commande $\mathcal{H}_\infty$ et μ-analyse : des outils pour la robustesse*, volume 2. Hermès, Paris.

Eberhart, R. and Shi, Y. (2001a). Particle Swarm Optimization : Developments, Applications and Resources. In *The IEEE Congress on Evolutionary Computation*, volume 1, pages 81– 86, Seoul.

Eberhart, R. and Shi, Y. (2001b). Tracking and optimizing dynamic systems with particle swarmss. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 94–100, Seoul.

Eberhart, R., Simpson, P., and Dobbins, R. (1996). *Computational Intelligence PC Tools*. Academic Press Professional, USA.

Ergezer, M., Simon, D., and Du, D. (2009). Oppositional biogeography-based optimization. In *The International Conference on Systems, Man and Cybernetics*, pages 1009–1014, Piscataway, New Jersey.

Farmer, J. D., Packard, N. H., and Perelson, A. S. (1986). The immune system, adaptation, and machine learning. *Physica D*, 1-3(2) :187–204.

Feng, G. and Lozano, R. (1999). *Adaptive Control Systems*. Newnes, Oxford.

Feo, T. A. and Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2) :67–71.

Fogel, L. J. (1962). Autonomous automata. *Industrial Research Magazine*, 4(2) :14–19.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization : Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, California.

Francis, B. A. and Zames, G. (1984). On $\mathcal{H}_\infty$ Optimal Sensitivity theory for SISO feedback systems. *IEEE Transactions on Automatic Control*, 29 :9–17.

Fraser, A. S. (1960). Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science*, 13(2) :150–162.

Gahinet, P. and Apkarian, P. (1994). A Linear Matrix Inequalities Approach to $\mathcal{H}_\infty$ Control. *Australian Journal of Biological Science*, 4(4) :421–448.

Galdos, G., Karimi, A., and Longchamp, R. (2011). RST Controller Design by Convex Optimization Using Frequency-Domain Data. In *The 18th IFAC World Congress*, pages 11429–11434, Milano.

Gangulya, S., Sahoob, N., and Dasc, D. (2013). Multiobjective particle swarm optimization based on fuzzy pareto dominance for possibilistic planning of electrical distribution systems incorporating distributed generation. *Fuzzy Sets and Systems*, 213 :47–73.

Garcia, G. (1999). *Contribution à la synthèse de lois de commande robuste par l'approche quadratique*. Habilitation à diriger des recherches, chapitre1.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5) :533–549.

Glover, F. (1989a). Tabu Search : Part I. *ORSA Journal on Computing*, 1(3) :190–206.

Glover, F. (1989b). Tabu Search : Part II. *ORSA Journal on Computing*, 2(3) :4–32.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Boston.

Goldberg, D., Korb, B., and Deb, K. (1989). Messy Genetic Algorithms : Motivation, Analysis, and First Results. *Complex Systems*, 3(5) :493–530.

Goldberg, D. E. (1994). *Algorithmes génétiques : exploration, optimisation et apprentissage automatique*. Addison-Wisley, France.

Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann.

Goldberg, D. E. and Richardson, J. (1987). Genetic Algorithm with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, Hillsdale, New Jersey.

Goss, S., Aron, S., Deneubourg, J. L., and Pasteels, J. M. (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12) :579–581.

Green, M. and Limebeer, D. J. (1995). *Robust Control*. Prentice-Hall, Englewood Cliffs.

Haggège, J., Bouallègue, S., and Benrejeb, M. (2009). Robust $\mathcal{H}_\infty$ Control Design for a DC Drive. *International Review of Automatic Control*, 2(4) :415–422.

Haggège, J. Y. (2003). *Sur la synthèse de processus complexes par des mèthodes neuro-floues*. Thèse de Doctorat, Ecole Nationale d'Ingénieurs de Tunis, Tunis.

Haggège, J. Y. and Benrejeb, M. (2000). Sur la synthèse d'un réguateur neuro-flou. In *Journée Scientifiques Franco-Tunisiennes JSFT'2000*, Monastir, Tunisie.

Haggège, J. Y., Benrejeb, M., and Borne, P. (2001a). , neuro-Fuzzy Control of a Bioreactor. In *The Smart Systems and Devices*, pages 398–403, Hammamet, Tunisia.

Haggège, J. Y., Benrejeb, M., and Borne, P. (2001b). A New Approach for On-Line Optimization of a Fuzzy Controller. In *The IEEE International Conference on Electronics, Circuits and Systems*, pages 971–975, Malta.

Hammouche, K., Diaf, M., and Siarry, P. (2010). A comparative study of various metaheuristic techniques applied to the multilevel thresholding problem. *Engineering Applications of Artificial Intelligence*, 23(5) :676–688.

Hastings, W. K. (1970). Monte-carlo sampling method using markov chains and their applications. *Biometrika*, 57(1) :676–688.

Herrero, J. M., Sanchis, J., and Martinez, M. (2008). A new graphical visualization of n-dimensional pareto front for decision-making in multiobjective optimization. *Information Science*, 178(20) :3908–3924.

Ho, S.-J., Ho, S.-Y., Hung, M.-H., Shu, L.-S., and Huang, H.-L. (2005). Designing Structure- Specified Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ Optimal Controllers Using an Intelligent Genetic Algorithm IGA. *IEEE Transactions on Control Systems Technology*, 13(6) :1119–1124.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA.

Hoque, M., Mukit, M. A., and Bikas, M. A. N. (2012). An Implementation of Intrusion Detection System Using Genetic Algorithm. *International Journal of Network Security and Its Applications*, 4(2) :109–120.

Horn, J. and Nafpliotis, N. (1993). Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical report, num : 93005, University of Illinois at Urbana Champaign, Urbana, Illinois, USA.

Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 82–87, Orlando.

Huang, V. L., Qin, A., Suganthan, P. N., and Tasgetiren, M. F. (2007). Multiobjective optimization based on self adaptive differential evolution algorithm. In *Proceedings of the Congress on Evolutionary Computation CEC' 2007*, pages 3601–2608, Singapore.

Iwasaki, T. and Skelton, R. E. (1994). All Controllers for the General $\mathcal{H}_\infty$ Control Problem : LMI Existence Conditions and State-Space Formulas. *Automatica*, 30(8) :1307–1317.

Jungjit, S. and Freitas, A. A. (2015). Lexicographic Multi-Objective Genetic Algorithm for Multi-Label Correlation-Based Feature Selection. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary*, pages 989–99, USA, New YorK.

Kennedy, J. (1998). The behavior of particles. In *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pages 581–589, San Diego, CA.

Kennedy, J. and Eberhart, R. (1995a). A new optimizer using particle swarm theory. In *The Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Indianapolis.

Kennedy, J. and Eberhart, R. (1995b). Particle Swarm Optimization. In *the IEEE International Joint Conference on Neural Networks*, pages 1942–1948, Perth.

Kennedy, J., Eberhart, R., and Shi, Y. (2001). *Swarm Intelligence*. Morgan Kaufmann Academic Press, USA.

Kim, K., Walewski, J., and Cho, Y. K. (2016). Multiobjective Construction Schedule Optimization Using Modified Niched Pareto Genetic Algorithm. *Journal of Management in Engineering*, 32(2) :1307–1317.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598) :671–680.

Koza, J. R. (1992). *Genetic Programming : On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems).* First edition, The MIT Press, London, England.

Kwakernaak, H. (1986). A polynomial approach to minimax frequency domain optimization of multivariable systems. *International Journal of Control*, 44(1) :117–156.

Kwakernaak, H. (1993). Robust Control and $\mathcal{H}_\infty$ Optimization : Tutorial Paper. *Automatica*, 29(2) :255–273.

Lampinen, J. and Zelinka, I. (1999). Mixed Integer-Discrete-Continuous Optimization by Differential Evolution - Part 1 : the optimization method. In *Proceedings of MENDEL'99- 5th International Mendel Conference on Soft Computing*, pages 71–76, Brno, Czech.

Landau, I. D. (1998). The RST digital controller design and applications. *Control Engineering Practice*, 6(2) :155–165.

Landau, I. D. (2002). *Commande des systèmes.* Hermès, Paris.

Landau, I. D. and Dugard, L. (1996). *Commande adaptative : aspects pratiques et théoriques.* Editions Masson, Paris.

Landau, I. D. and Karimi, A. (1998). Robust Digital Control using Pole placement with Sensitivity function shaping method. *Control Engineering Practice*, 8(2) :191–210.

Landau, I. D., Lozano, R., and M'Saad, M. (1998). *Adaptive Control.* Springer-Verlag, London.

Langer, J. (1998). *Synthèse de régulateurs numériques robustes : Application aux structures souples.* Thèse de Doctorat, Institut National Polytechnique de Grenoble, Grenoble.

Langer, J. and Landau, I. D. (1999). Combined pole placement/sensitivity function shaping method using convex optimization criteria. *Automatica*, 35(2) :1111–1120.

Larminat, P. (2007). *Analysis and Control of Linear Systems.* ISTE, London.

Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002a). Archiving with guaranteed convergence and diversity in Multi-Objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 439–447, San Francisco, California.

Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002b). Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3) :263–282.

Lee, C. C. (1990a). Fuzzy Logic in Control systems : Fuzzy Logic Controller-Part I. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2) :404–418.

Lee, C. C. (1990b). Fuzzy Logic in Control systems : Fuzzy Logic Controller-Part II. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2) :419–435.

Lopez-Ibanez, M., Stutzle, T., and Dorigo, M. (2015). Ant Colony Optimization : A Component-Wise Overview. Technical report, IRIDIA.

Ma, H. and Simon, D. (2011). Blended biogeography-based optimization for constrained optimization. *Engineering Applications of Artificial Intelligence*, 24(3) :517–525.

MacArthur, R. and Wilson, E. (1967). *The Theory of Biogeography*. Princeton University Press, Princeton, New Jersey.

Maciejowski, J. M. (1989). *Multivariable Feedback Design*. Addison-Wesley, New York.

Madavan, N. K. (2002). Multiobjective optimization using a Pareto differential evolution approach. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1145–1150, New Jersey.

Madiouni, R., Bouallègue, S., Haggège, J., and Siarry, P. (2013). Particle Swarm Optimization-based Design of Polynomial RST Controllers. In *Proceedings of the 10th International Multi-Conference on Systems, Signals and Devices,*, Hammamet, Tunisia.

Madiouni, R., Bouallègue, S., Haggège, J., and Siarry, P. (2016). Robust RST Control Design based on Multi-Objective Particle Swarm Optimization Approach. *International Journal of Control, Automation, and Systems*, 14(6) :1–11.

Mamdani, E. H. (1974). Application of fuzzy logic algorithms for control of simple dynamic plant. *Institute of Electrical Engineering IEE*, 121(12) :1585–1588.

Martinez, S. Z. and Coello.Coello, C. A. (2011). A Multiobjective Particle Swarm Optimizer Based on Decomposition. In *The 13th annual Genetic and Evolutionary Computation Conference*, pages 69–76, Dublin.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6) :1087–1092.

Miranda, V. and Fonseca, N. (2002). New evolutionary particle swarm algorithm applied to voltage/VAR control. In *Proceedings of the 14th Power Systems Computation Conference*, pages 1–6, Seville.

Moslemi, H. and Zandieh, M. (2011). Comparisons of some improving strategies on MOPSO for multi-objective (r, Q) inventory system. *Expert Systems with Applications*, 38(10) :12051–12057.

Mostaghim, S. and Teich, J. (2003). Strategies for finding good local guides in multi-objective particle swarm optimization. In *The IEEE Swarm Intelligence Symposium*, pages 26.33, DOI :10.1109/SIS.2003.1202243, Seville.

Muhlenbein, H. and Paass, G. (1996). From recombination of genes to the estimation of distributions i. Binary parameters. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, Springer-Verlag*, pages 178–187, London, UK.

Murray-Smith, R. and Johansen, T. A. (1997). *Multiple model approaches to modeling and control.* Taylor and Francis, London.

Nakib, A., Oulhadj, H., and Siarry, P. (2010). Image thresholding based on Pareto multiobjective optimization. *Engineering Applications of Artificial Intelligence*, 23(3) :313–320.

Nelder, J. A. and Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, 7(4) :308–313.

Nerrand, O., Roussel-Ragot, P., Personnaz, L., Dreyfus, G., and Marcos, S. (1992). Neural networks and nonlinear adaptive filtering : unifying concepts and new algorithms. *Neural Computation*, 5(2) :165–199.

Nocedal, J. and Wright, S. J. (2000). *Numerical Optimization.* Springer, New York.

Okamoto, Y. and Hansmann, U. H. (1995). Thermodynamics of helix-coil transitions studied by multicanonical algorithms. *The journal of physical chemistry*, 82(99) :11276–11287.

Oustaloup, A. (1975). *Etude et réalisation d'un système d'asservissement d'ordre 3/2 de la fréquence d'un laser à colorant continu.* Thèse de Doctorat, Université de Bordeaux I, Bordeaux.

Oustaloup, A. (1995). *La dérivation non entière : théorie, synthèse et applications.* Hermès, Paris.

Oustaloup, A., Moreau, X., and Mathieu, B. (1999). *Commande CRONE : principes et exemples d'application.* Techniques de l'Ingénieur, Traité Informatique Industrielle, Paris.

Ozcan, E. and Mohan, C. K. (1999). Particle Swarm Optimization : surfing the waves. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1939–1944, Washington.

Passino, K. M. and Yurkovich, S. (1998). *Fuzzy Control.* Addison-Wesley, California.

Peer, E. S., Engelbrecht, A. P., and Van-Den-Bergh, F. (2003). Using neighborhoods with the guaranteed convergence PSO. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS'03)*, pages 235–242, Indiana.

Polak, E. (1997). *Optimization : Algorithms and Consistent Approximations.* Springer-Verlag, New York.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle Swarm Optimization : An Overview. *Swarm Intelligence*, 1(1) :33–57.

Postlethwaite, I., Tsai, M. C., and Gu, D. W. (1990). Weighting function selection in $\mathcal{H}_\infty$ design. In *Proceedings of the 11th IFAC World Congress*, volume 5, pages 104–109, Tallin, Estonia.

Price, K. V., Storn, R. M., and Lampinen, J. A. (2005). *Differential Evolution : A Practical Approach to Global Optimization.* Natural Computing Series, Springer-Verlag, Berlin, Germany.

Prochazka, H. and Landau, I. D. (2002). Logiciel pour l'enseignement et le calcul du placement de pôles robuste. In *the Conférence Internationale Francophone d'Automatique*, volume 5, pages 694–698, Nantes.

Prochazka, H. and Landau, I. D. (2003). Multivariable robust controller design by pole placement and sensitivity shaping. Technical report, Laboratoire d'Automatique de Grenoble.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Technical report, Library translation 1122, Ministry of Aviation, Royal Air Force Establishment (UK).

Reyes-Sierra, M. and Coello.Coello, C. A. (2006). Multi-Objective Particle Swarm Optimizers : A Survey of the State-of-the Art. *International Journal of Computational Intelligence Research*, 2(3) :287– 308.

Richalet, J. (1993). Industrial Applications of Model Based Predictive Contro. *Automatica*, 29(5) :1251–1274.

Richalet, J., Rault, A., Testud, J. L., and Papon, J. (1976). Algorithmic control of industrial processes. In *Proceedings of 4th IFAC Symposium on Identication and System Parameter Estimation*, pages 1119–1167, Tbilissi.

Richalet, J., Rault, A., Testud, J. L., and Papon, J. (1978). Model Predictive Heuristic Control : Applications to Industrial Processes. *Automatica*, 14(5) :413–428.

Robic, T. and Filipic, B. (2005). DEMO : Differential Evolution for Multiobjective Optimization. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 520–533, Guanajuato.

Robinsonand, J., Sinton, S., and Rahmat-Samii, Y. (2002). Particle Swarm, Genetic Algorithm, and Their hybrids : Optimization of a Profiled Corrugated Horn Antenna. In *Proceedings of the IEEE International Symposium on Antennas and Propagation*, pages 314–317, San Antonio, Texas.

Rotea, M. A. and Khargonekar, P. P. (1991). Generalized $\mathcal{H}_2/\mathcal{H}_\infty$ Control via Convex Optimization. In *the 30th Conference on Decision and Control*, pages 2719–2720, Grighton England.

Rotella, F., Carillo, F., and Ayadi, M. (2001). Polynomial controller design based on flatness. In *the First IFAC-IEEE Symposium on System Structure and Control*, pages 936– 941, Prague.

Rotella, F. and Fourquet, J. Y. (2001). *Commande des systhèmes linéaires à plusieurs entrée.* Notes de cours, Ecole Nationale d'Ingénieurs de Tarbes, Tarbes.

Salazar-Lechuga, M. and Rowe, J. E. (2005). Particle Swarm Optimization and Fitness Sharing to solve Multi-Objective Optimization Problems. In *the 2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, pages 1204–1211, Edinburgh, Scotland.

Salima, M. (2009). *Commande adaptative et précatives de la machine asynchrone.* Thèse de Doctorat, Faculté des sciences de l'Ingénieur, Université Mentouri de Constantine, Constantine.

Schaffer, J. D. (1985). Multi-objective optimization with vector evaluated genetic algorithm. In *Proceedings of the first International Conference on Genetic Algorithms*, pages 93–100, Pittsburgh, USA.

Schrijver, A. (1998). *Theory of Linear and Integer Programming.* John Wiley and Sons, New York.

Sheloka, P., Siarry, P., Jayaraman, V., and Kulkarni, B. (2007). Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, 188(1) :129–142.

Shi, Y. and Eberhart, R. (1998a). A modified Particle Swarm Optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69–73, Piscataway.

Shi, Y. and Eberhart, R. (1998b). Parameters Selection in Particle Swarm Optimization. In *Proceedings of the 7th Annual Conference on Evolutionary Programming*, pages 591–600, New YorK.

Shi, Y. and Eberhart, R. (1999). Empirical study of Particle Swarm Optimization. In *the Congress on Evolutionary Computation*, pages 1945–1950, Washington.

Shi, Y., Eberhart, R., and Chen, Y. B. (1997). Implementation of an Evolutionary Fuzzy Epert System. *IEEE Transactions on Fuzzy Systems*, 7(2) :109–119.

Shokrian, M. and Highb, K. A. (2014). Application of a multi objective multi-leader particle swarm optimization algorithm on NLP and MINLP problems. *Computers and Chemical Engineering*, 60(2) :57–75.

Siarry, P. and Dreyfus, G. (1989). *La méthode du recuit simulé : théorie et applications.* ESPCI-IDSET, Paris.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, 12(6) :702–713.

Srinivas, N. and Deb, K. (1993). Multiobjective optimization using nondominated sorting ingenetic algorithms. Technical report, Kanput, India.

Storn, R. M. and Price, K. V. (1997). Differential Evolution : a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4) :341–359.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1) :166–172.

Talbi, E. G. (1999). *Métaheuristiques pour l'optimisation combinatoire multi-objectif : Etat de l'art.*

Tasan, A. S. and Gen, M. (2012). A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers and Industrial Engineering*, 62 :755–761.

Tfaili, W. (2012). *Conception d'un algorithme de colonie de fourmis pour l'optimisation continue dynamique.* Thèse de Doctorat, Université Paris 12 Val de Marne, Paris.

Thierry, B. (2000). *Modélisation Expérimentale des Systèmes dynamiques Interconnectés.* Habilitation à diriger des recherches, Université Henri Poincaré, Nancy.

Trelea, I. C. (2003). TheParticle Swarm Optimization algorithm : convergence analysis and parameter selection. *Information Processing Letters*, 85(6) :317–325.

Van-Den-Bergh, F. (2001). *An analysis of Particle Swarm Optimizers.* PhD thesis, University of Pretoria, South Africa.

Van-Veldhuizen, D. A. (1999). *Multiobjective evolutionary algorithms : Classifications, analyzes, and new innovations.* Ph.D. dissertation, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson.

Van-Veldhuizen, D. A. and Lamont, G. B. (2000a). Multiobjective Optimization with Messy Genetic Algorithms. In *Proceedings of the Symposium on Applied Computing*, pages 470–476, Italy.

Van-Veldhuizen, D. A. and Lamont, G. B. (2000b). On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the Congress on Evolutionary Computation (CEC'2000)*, volume 1, pages 204–211, New Jersey.

Varga, A. (2000). Robust pole assignment via Sylvester equation based state feed-back parametrization. In *Proceedings of the IEEE International Symposium on Computer Aided Control System Design (CACSD'2000)*, pages 13–18, Anchorage, Alaska.

Walke, A., Hallam, J., and Willshaw, D. (1993). Beehavior in a Mobile Robot : The Construction of a Self-Organized Cognitive Map and its Use in Robot Navigation within a Complex, Natural Environment. In *International Conference on Neural Networks*, volume 3, pages 1451–1456, Piscataway, NJ.

Whitaker, H. P., Yamron, J., and Kezer, A. (1958). Design of modelreference adaptive control systems for aircraft. Technical report, Instrumentation Laboratory, MIT. Cambridge.

Xiaobing, Y., Mei, C., and Jie, C. (2015). A novel mutation differential evolution for global optimization. *Journal of Intelligent and Fuzzy Systems*, 28(3) :1047–1060.

Xue, F., Sanderson, A. C., and Graves, R. J. (2003). Pareto-based multi-objective differential evolution. In *Proceedings of the Congress on Evolutionary Computation (CEC'2003)*, volume 2, pages 862–869, Canberra.

Youla, D. C., Jabri, H. A., and Bongiorno, J. (1976). Modern Wiener-Hopf design of optimal controllers, Part II : The multivariable case. *IEEE Transactions on Automatic Control*, 3(21) :319–338.

Zadeh, L. (1965). Fuzzy Sets. *Information Control*, 8(3) :338–353.

Zames, G. (1981). Feedback and Optimal Sensitivity : Model Reference Transformations, Multiplicative Semi Norms, and Apprximate Inverses. *IEEE Transactions on Automatic Control*, 26(2) :301–320.

Zames, G. and Francis, B. A. (1983). Feedback, minimax sensitivity, and optimal robustness. *IEEE Transactions on Automatic Control*, 28 :585–601.

Zhang, C., Ning, J., Lu, S., Ouyang, D., and Ding, T. (2009). A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. *Operations Research Letters*, 37(2) :117–122.

Zhao, S., Nagaratnam-Suganthan, P., and Zhang, Q. (2012). Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes. *IEEE Transactionson Evolutionary Computation*, 16(3) :442–446.

Zhaoa, Z., Yanga, J., Hua, Z., and Chea, H. (2016). A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric latin hypercube design for unconstrained optimization problems. *European Journal of Operational Research*, 250(1) :30–45.

Zhengchao, L., Xudong, Z., and YJinyong, Y. (2016). On robust control of continuous-time systems with state-dependent uncertainties and its application to mechanical systems. *ISA Transactions*, 60 :12–20.

Zhou, K. and Doyle, J. C. (1998). *Essentials of Robust Control*. Prentice-Hall, New Jersey.

Zhou, K., Doyle, J. C., and Glover, K. (1996). *Robust and Optimal Control*. Prentice-Hall, New Jersey.

Zito, G. (2005). *Contribution à une méthodologie intégrée d'identification et commande des systèmes industriels*. Thèse de Doctorat, Institut National Polytechnique de Grenoble, Grenoble.

Zitzler, E., Thiele, L., and Bader, J. (2010). On set-based multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 14(1) :58–79.

Zydallis, J. B., Van-Veldhuizen, D. A., and Lamont, G. B. (2001). A statistical Comparison of Multiobjective Evolutionary Algorithms Including the MOMGA-II. In *E. Zitzler,*

K. Deb, L. Thiele, C.A. Coello Coello, and D. Corne (Editors), First International Conference on Evolutionary Multi-Criterion Optimization, volume 1993, pages 226–240.