N° d'ordre : 2011REN1E009 ANNÉE 2011

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention : Traitement du signal et télécommunications
Ecole doctorale MATISSE

présentée par

Hai-Nam NGUYEN

préparée à l'unité de recherche UMR 6074
Équipe-Projet Inria CAIRN à Lannion
École Nationale Supérieure des Sciences Appliquées et de Technologie

---

Optimisation de la précision de calcul pour la réduction d'énergie des systèmes embarqués

Jury composé de :

Michel Jezequel
Professeur à Télécom Bretagne / *rapporteur*
Laurent Fesquet
Maître de conférences HDR à INPG / *rapporteur*
François Horlin
Professeur à l'Université Libre de Bruxelles / *examinateur*
Dominique Noguet
Ingénieur – Chercheur au CEA LETI / *examinateur*
Christophe Moy
Professeur à Supélec / *examinateur*
Daniel Ménard
Maître de conférences HDR à l'Univ. de Rennes 1 / *Encadrant*
Olivier Sentieys
Professeur à l'Université de Rennes 1 / *Directeur de thèse*

Remerciements

Mon travail de thèse, le plus beau fruit de ma vie étudiante, n'aurait pas parvenir son objectif sans le soutien de mon entourage.

Je tiens à remercier, en tout premier lieu, mon directeur de thèse, monsieur Olivier Sentieys, chef du projet CAIRN, de l'accueil qu'il m'a réservé au sein de son équipe et de la confiance qu'il m'a témoignée pendant mes longues années de recherches. C'est grâce à lui que j'ai découvert le monde des Mac. Tout au long de ces années, il a su m'orienter aux bons moments tout en me laissant la liberté dans l'organisation de mon travail.

Je remercie également monsieur Daniel Ménard pour la gentillesse et la patience qu'il a témoignées à mon égard durant mes recherches et les relectures de cette thèse. Malgré son planning parfois surchargé, il a toujours été disponible pour le suivi quotidien et d'intéressantes discussions. Ses conseils et ses propositions m'ont aidé éclaircir mon sujet de thèse.

J'adresse mes sincères remerciements à messieurs Michel Jezequel, professeur à Télécom Bretagne et Laurent Fesquet, maître de conférences à INPG, chef du groupe CIS/TIMA pour avoir accepté de juger mon travail en tant que rapporteurs.

Je remercie également messieurs François Horlin, professeur à l'Université Libre de Bruxelles, Dominique Noguet, ingénieur – chef du groupe LCS/ANP au CEA et Christophe Moy, professeur à Supélec de leur participation au jury de thèse.

Je remercie le Conseil Général des Côtes d'Armor pour m'a avoir soutenu financièrement dans la réalisation de cette thèse.

Je voudrais témoigner ma reconnaissance à toute l'équipe projet CAIRN qui m'a accueilli avec amitié. Grâce à elle, j'ai pu commencer mes recherches dans un domaine que je n'ai pas tout à fait connu auparavant.

Un grand merci à mes amis qui étaient et qui sont à Lannion, que je ne listerai pas pour ne faire oublier personne. Avec eux, j'ai passé de bons moments de repos et de sport (presque) tous les week-ends.

Je réserve, enfin, une dédicace spéciale à ma femme qui a toujours été présente à mes côtés, à mon frère et à mes parents au Vietnam pour leur soutien moral.

Table des matières

Table des matières iii

Table des figures vii

Liste des tableaux xi

Introduction 1

1 Conversion en virgule fixe 7
1.1 Introduction . . . 7
1.2 Codage des données . . . 7
1.2.1 Représentation des nombres entiers . . . 7
1.2.2 Codages des nombres réels . . . 8
1.3 Arithmétique virgule fixe . . . 10
1.3.1 Opérations arithmétiques en virgule fixe . . . 10
1.3.2 Règles de codage en virgule fixe . . . 11
1.3.3 Implantation . . . 13
1.4 Méthodologie de conversion en virgule fixe . . . 13
1.4.1 Évaluation de la dynamique . . . 13
1.4.2 Optimisation de la largeur des données . . . 17
1.4.3 Conclusions . . . 20

I Adaptation dynamique des largeurs 21

2 Méthodologie d'adaptation dynamique de la précision 25
2.1 Introduction . . . 25
2.2 Méthodes d'adaptation existantes . . . 26
2.2.1 Adaptation en fonction de la qualité de service désirée . . . 26
2.2.2 Adaptation en fonction du mode de fonctionnement . . . 26
2.2.3 Adaptation en fonction des performances de l'application . . . 26
2.3 Principe de l'adaptation dynamique de la précision . . . 27
2.3.1 Métriques p . . . 28
2.3.2 Contraintes de performance . . . 28
2.4 Support d'exécution . . . 29
2.4.1 Granularité en termes de largeur des données . . . 29

2.4.2 Définition du support d'exécution 35
2.5 Conclusion 36

3 Adaptation dynamique : systèmes de communication sans-fil 37
3.1 Chaîne de transmission QPSK 37
3.1.1 Introduction 37
3.1.2 Estimation de la dynamique 38
3.1.3 Analyse de la précision 38
3.1.4 Conclusion 43
3.2 Systèmes à étalement de spectre DS-CDMA 43
3.2.1 Introduction 43
3.2.2 Récepteur en râteau *(Rake Receiver)* 46
3.2.3 Estimation de la dynamique 49
3.2.4 Évaluation de précision 50
3.2.5 Conclusion 52
3.3 Adaptation du récepteur WCDMA 53
3.3.1 Présentation du système WCDMA 53
3.3.2 Décodeur de symbole 54
3.3.3 Module de recherche de trajets 60
3.3.4 Scénario réel 67
3.4 Conclusions 69

II Optimisation des largeurs 71

4 État de l'art sur une classe d'algorithmes d'optimisation combinatoire 75
4.1 Introduction : rappel du problème à résoudre 75
4.1.1 Contexte du problème d'optimisation 76
4.1.2 Classification des algorithmes 77
4.1.3 Définitions 79
4.2 Algorithmes d'optimisation combinatoire 79
4.2.1 Algorithmes déterministes 80
4.2.2 Algorithmes aléatoires 81
4.3 Optimisation des largeurs 83
4.3.1 Algorithmes déterministes 83
4.3.2 Algorithmes aléatoires 86
4.3.3 Bilan 87
4.4 Conclusions 88

5 Améliorations de la technique basées sur les algorithmes génétiques 91
5.1 Présentation des algorithmes génétiques 91
5.1.1 Introduction aux algorithmes évolutionnistes 91
5.1.2 Algorithmes génétiques 96
5.1.3 Algorithmes génétiques multi-objectifs 99
5.1.4 Applications des algorithmes génétiques à l'optimisation des largeurs 102
5.2 Analyse qualitative des algorithmes génétiques 103
5.2.1 Avantages 104
5.2.2 Inconvénients 105
5.2.3 Conclusion 106

5.3 Proposition d'un algorithme amélioré . . . 106
5.3.1 Contexte . . . 106
5.3.2 Extensions des algorithmes génétiques . . . 107
5.3.3 Proposition d'une version améliorée de MOGA . . . 108
5.3.4 Expérimentations . . . 110
5.4 Conclusions . . . 115

6 Recherche locale stochastique 117
6.1 Recherche avec tabous . . . 117
6.1.1 Introduction . . . 117
6.1.2 Application au problème d'optimisation des largeurs . . . 118
6.1.3 Expérimentations et résultats . . . 121
6.1.4 Conclusions . . . 121
6.2 Procédures de recherche gloutonne aléatoire et adaptative (GRASP) . . . 124
6.2.1 Algorithmes de recherche locale stochastique . . . 124
6.2.2 Algorithme de recherche avide aléatoire adaptative . . . 125
6.2.3 Avantages et inconvénients . . . 126
6.3 Application de GRASP à l'optimisation des largeurs . . . 127
6.3.1 Phase de construction . . . 127
6.3.2 Phase de recherche locale . . . 128
6.3.3 Discussion . . . 128
6.4 Évaluation de GRASP pour l'optimisation des largeurs . . . 128
6.4.1 Algorithme GRASP–a . . . 129
6.4.2 Algorithme GRASP–ac . . . 133
6.4.3 Combinaison de GRASP–a et GRASP–ac . . . 137
6.5 Conclusions . . . 137

Conclusions et perspectives 139

A Familles des algorithmes évolutionnistes 143
A.1 Stratégies d'évolution . . . 144
A.2 Programmation évolutionniste . . . 145
A.3 Algorithmes génétiques . . . 146

B Bibliothèque des opérateurs utilisée pour les simulations 147
B.1 Registres . . . 147
B.2 Additionneurs . . . 147
B.3 Multiplieurs . . . 147

Bibliographie 149

Table des figures

1.1 Codage en virgule fixe 9
1.2 Codage en virgule flottante normalisée. 9

2.1 Principe de l'approche d'adaptation dynamique de la précision 27
2.2 Granularité en termes de largeur des données 29
2.3 opérateur supportant le parallélisme de données 30

3.1 Graphe flot d'une chaîne de transmission QPSK. 38
3.2 Analyse de la dynamique pour la détection de symboles dans le cas d'un canal BBGA et la modulation QPSK. 39
3.3 Valeur théorique du TEB d'une modulation QPSK avec un canal BBGA pour différentes quantifications et différents RSB. 40
3.4 Largeur théorique de la partie fractionnaire d'une modulation QPSK avec un canal BBGA pour différents critères de précision et différents RSB. . . . 41
3.5 TEB simulé pour une modulation QPSK avec un canal BBGA pour différentes quantifications et différents RSB. 42
3.6 La technique d'étalement de spectre par séquence directe. La période du *chip* (T_c) est beaucoup plus faible que la période du symbole (T_s). 45
3.7 Principe d'un système de transmission utilisant la technique DSSS. Le signal est modulé en BPSK et l'étalement de spectre est fait avec une séquence pseudo–aléatoire binaire. 45
3.8 Récepteur en râteau de type MRC. τ_i vient du module de recherche de trajet et $\hat{\alpha}_i(t)$ vient de l'estimateur de canal. 48
3.9 Distribution de la puissance d'émission du récepteur mobile à 64 kbps avec une probabilité de coupure de 5% selon [73]. 54
3.10 Graphe flot de données d'un décodeur de symboles dans un récepteur WCDMA. 55
3.11 Valeur estimée et valeur réelle de la dynamique dans un décodeur pour différents RSB. 57
3.12 Résultats de simulation du TEB dans les conditions de canal différentes pour le décodeur de symbole. 58
3.13 Puissance du signal et du bruit en fonction du RSB. 58
3.14 Consommation d'énergie normalisée du décodeur sur une cible ASIC. 59
3.15 Consommation d'énergie normalisée du module de décodage dans le cas d'une implantation avec des opérateurs SWP. 60

3.16 Énergie consommée pour le module de décodage pour différents facteurs d'étalement (SF) 61
3.17 Graphe flot de données d'un module de recherche de trajet PDP (*Power Delay Profile*) 61
3.18 Dynamique au sein du module de recherche de trajets obtenue par simulation et par la méthode analytique. 62
3.19 Une instance du profil de puissance pour un canal de Rayleigh de la norme 3GPP cas 3. SNR à 10 dB et α fixé à 7. 63
3.20 Comparaison du taux de fausse alarme et taux de non-détection sur différents seuils : $\alpha = 6$ et $\alpha = 7$. Le niveau de bruit de quantification varie entre -70 dB à -90 dB. 64
3.21 Taux de FA et taux de MD avec $\alpha = 7$. Le niveau de bruit de quantification varie entre -60 dB à -80 dB. 65
3.22 Consommation d'énergie pour le module de recherche de trajets. 66
3.23 Consommation d'énergie pour le module de recherche de trajets avec la technique SWP. 66
3.24 Variation du RSB lors du déplacement d'un utilisateur dans une cellule. . . 68

5.1 Graphe flot d'un algorithme évolutionniste 92
5.2 Notations utilisées pour l'étude des algorithmes évolutionnistes 93
5.3 Codage d'un chromosome dans les algorithmes génétiques 96
5.4 Le codage position réelle versus le code Gray pour la représentation de chromosome. Le code Gray permet de minimiser la distance Hamming entre les individus proches, de minimiser la perturbation et est souvent meilleur [71] que le codage positionnel dans les algorithmes génétiques. 97
5.5 Deux méthodes pour classer les individus sur les frontières de Pareto. 101
5.6 Comparaison entre l'algorithme génétique et les recherches locales sur un filtre IIR à 7 variables. Ces frontières Pareto sont obtenues aux $50^{\text{ème}}$ et $500^{\text{ème}}$ générations, sur une population comprenant 90 individus. Les algorithmes gloutons utilisent comme une métrique de sélection de la direction de déplacement la précision (DM), le coût (CM) ou une combinaison des deux (CDM). Source : [59] 104
5.7 Frontières de Pareto obtenues pour un filtre IIR composé de 18 variables. Les résultats sont présentés pour différentes générations (machine : PC1). . 111
5.8 Efficacité de l'élitisme par rapport au nombre de générations. Le résultat est plus marqué quand le nombre de générations est faible. Le nombre d'individus sur la frontière de Pareto augmente naturellement avec le nombre de générations. Les résultats sont présentés pour une FFT–64 composée de 8 variables à optimiser(machine : PC1). 112
5.9 Efficacité de l'élitisme. Simulation sur un filtre NLMS 128 entrées avec 49 variables. La conservation totale des élites permet d'augmenter la diversité et d'avoir une meilleure solution dans certains intervalles de précision : p. ex. à la $500^{\text{ème}}$ génération, pour un RSBQ de 37 dB à 42 dB (machine : PC1).113

6.1 Comparaison des performances de l'algorithme glouton et de l'algorithme de recherche avec tabous sur les filtres IIR. L'écart-type représente la variation du résultat en fonction du critère de précision (machine : PC2). 122

6.2 Comparaison des performances de l'algorithme glouton et de l'algorithme de recherche avec tabous sur les FFT. L'écart-type représente la variation du résultat en fonction du critère de précision (machine : PC2). 123
6.3 Comparaison des performances de l'algorithme glouton et de l'algorithme de recherche avec tabous sur les filtres NLMS. L'écart-type représente la variation du résultat en fonction du critère de précision (machine : PC2). . 123
6.4 Énergie en 10^{-9} J obtenue pour différents algorithmes d'optimisation (glouton–a, glouton–ac, CDM, MOGA et GRASP–a) sur plusieurs configurations des filtres IIR pour une contrainte de précision de P_b fixée à $P_{b,\max} = -55$ dB (machine : PC1). 132
6.5 Temps d'exécution en seconde pour différents algorithmes d'optimisation (glouton–a, glouton–ac, CDM, MOGA et GRASP–a) sur plusieurs configurations des filtres IIR. Dans ces trois configurations, le temps d'optimisation des algorithmes génétiques reste le même. Les algorithmes de recherche locale nécessite plus de temps quand la taille du problème augmente (machine : PC1). 133
6.6 Comparaison des performances de différents algorithmes d'optimisation (glouton–a, tabou, glouton–ac et GRASP–a) sur un filtre IIR pour différents nombres de variables et pour différents critères de précision. Dans la plupart des cas, la meilleure solution est obtenue par l'algorithme GRASP–a (machine : PC2).134
6.7 Comparaison des performances de deux algorithmes d'optimisation glouton–ac et GRASP–ac sur une FFT pour différents nombres de variables et pour différents critères de précision (machine : PC2). 135
6.8 Comparaison des performances de deux algorithmes d'optimisation glouton–ac et GRASP–ac sur un NLMS pour différents nombres de variables et pour différents critères de précision (machine : PC2). 136
6.9 Performance des algorithmes d'optimisation des largeurs à travers de trois familles d'application : IIR, FFT et NLMS. Les résultats sont normalisés : 100% est la meilleure solution. La combinaison GRASP–a/ac obtient toujours la meilleure performance. 137

A.1 La recombinaison de programme exécutable présenté en arbre. Source : [8] . 143

B.1 Consommation d'énergie des registres dans notre bibliothèque. 147
B.2 Consommation d'énergie des additionneurs dans notre bibliothèque. 148
B.3 Consommation d'énergie des multiplieurs dans notre bibliothèque. 148

Liste des tableaux

2.1 Jeu d'instructions du TMS320C64x+ utilisant la technique SWP 32

3.1 Largeur optimale d'un récepteur en râteau obtenue pour différents rapports signal sur bruit. 59
3.2 Largeurs optimisées du module de recherche de trajets en fonction du RSB. 65
3.3 Consommation d'énergie (Joule) du récepteur WCDMA 68

4.1 Tableau récapitulatif des principales méthodes d'optimisation utilisées dans la littérature 88

5.1 Comparaison de deux algorithmes génétiques sur un filtre IIR–8 (18 variables). n_g représente le nombre de générations. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 114
5.2 Comparaison de deux algorithmes génétiques sur un filtre IIR–8 (36 variables). n_g représente le nombre de générations. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 114
5.3 Comparaison de deux algorithmes génétiques sur un filtre NLMS 128 entrées, (49 variables). n_g représente le nombre de générations. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 115

6.1 Comparaison de l'algorithme glouton et de la recherche avec tabous sur les résultats et le temps d'exécution. Trois applications IIR, FFT et NLMS sont configurées sur différents nombres de variables (machine : PC2). 122
6.2 Comparaison de différents algorithmes d'optimisation sur un filtre IIR–8 (18 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 129
6.3 Comparaison de différents algorithmes d'optimisation sur un filtre IIR–8 (36 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 130

6.4 Comparaison de différents algorithmes d'optimisation sur une FFT 64 (12 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 130
6.5 Comparaison de différents algorithmes d'optimisation sur une FFT 64 (20 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 131
6.6 Comparaison de différents algorithmes d'optimisation sur un NLMS 128 (25 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 131
6.7 Comparaison de différents algorithmes d'optimisation sur un NLMS 128 (49 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1). 132

Introduction

Contexte de l'étude

L'évolution des technologies des circuits intégrés permet d'implanter au sein des systèmes embarqués de nouvelles applications, plus complexes, pour le traitement du signal, de l'image et de la vidéo. En particulier, le secteur des télécommunications sans fil intègre de nombreuses applications de traitement numérique du signal (TNS). En 2008, 74% des processeurs de traitement du signal *(Digital Signal Processor - DSP)* ont été vendus pour les applications sans fil [142]. Ces terminaux sans fil sont majoritairement des produits nomades et sont donc alimentés par batterie. La maîtrise de la consommation d'énergie est l'un des challenges majeurs de la conception de ces terminaux. De nouveaux services sont fournis (image, vidéo, accès Internet) et nécessitent des débits de données toujours plus élevés. Par conséquent, la complexité de la partie numérique de traitement en bande de basse augmente. Toutefois, la consommation d'énergie ne peut pas augmenter en raison de l'autonomie de la batterie. Ainsi, de nouvelles stratégies visant à réduire ou à maintenir à un niveau raisonnable la consommation d'énergie doivent être proposées.

Ces applications orientées calcul sont implantées dans les systèmes embarqués à l'aide de l'arithmétique virgule fixe afin de satisfaire les contraintes de coût, de consommation d'énergie inhérentes à ces systèmes [37, 53]. Comme la largeur [1] des bus et des mémoires au sein des architectures en virgule fixe est plus faible, le prix et la consommation d'énergie de ces architectures sont moins importants. L'arithmétique en virgule flottante basée sur la norme IEEE-754 nécessite d'utiliser des données codées sur au moins 32 bits. A titre d'exemple, la majorité des DSP virgule fixe possède une largeur naturelle nettement plus faible (16 bits). De plus, les opérateurs utilisant l'arithmétique à virgule fixe étant moins complexes, les architectures travaillant uniquement en virgule fixe sont plus rapides. Le sur-coût en termes de temps d'exécution, lié à l'émulation de l'arithmétique à virgule flottante au sein d'architectures en virgule fixe étant élevé (facteur 10 à 500 [148]), il est nécessaire de coder l'ensemble des données en virgule fixe.

1. Nombre de bits.

Problématique

La réduction du temps de mise sur le marché des applications exige l'utilisation d'outils de développement de haut niveau permettant d'automatiser certaines tâches. Le codage manuel des données est une tâche fastidieuse et source d'erreurs. Certaines expérimentations [55] ont montré que le codage manuel des données peut représenter jusqu'à 30% du temps de développement d'une application. Ainsi, des méthodologies de codage automatique des données en virgule fixe et les outils associés sont nécessaires car le codage manuel se révèle être un frein important à la diminution du temps de conception. Ces outils de conversion automatique en virgule fixe ont un double objectif. Ils permettent d'accélérer le développement des applications mais aussi d'optimiser le coût de l'implantation.

La problématique est de rechercher la meilleure adéquation algorithme architecture d'un point de vue du codage des données en virgule fixe. L'utilisation de l'arithmétique à virgule fixe conduit à une dégradation de la précision des calculs réalisés. Ainsi, le codage des données doit fournir une précision des calculs suffisante pour satisfaire les contraintes de qualité associées à l'application et doit permettre d'obtenir une implantation efficace au sein de l'architecture cible. Les meilleures solutions sont obtenues lorsqu'une largeur propre à chaque opérateur ou groupe d'opérations peut être choisie. La conversion en virgule fixe manuelle ne permet pas d'explorer l'espace de conception. En effet, le développeur ne peut tester que quelques largeurs différentes. Le cas échéant, une des solutions consiste à choisir une largeur identique pour toutes les opérations et la largeur minimale respectant la contrainte de précision est retenue. Quant à de la conversion automatique en virgule fixe, un processus d'optimisation des largeurs est mis en œuvre afin de minimiser le coût de l'implantation sous contrainte de précision. L'exploration de l'espace de conception nécessite de posséder une approche d'évaluation de la précision efficace en termes de temps d'exécution. En particuliers, les approches analytiques sont privilégiées par rapport aux approches par simulation. Le processus d'optimisation étant composé d'un nombre de variables à optimiser pouvant être important, il est nécessaire d'avoir un algorithme d'optimisation efficace en termes de temps d'exécution et de qualité de la solution trouvée. Dans le cadre de nos travaux de recherche, le coût de l'implantation est évalué à travers la consommation d'énergie.

Une des étapes importantes de la conversion en virgule fixe est la détermination de la contrainte de précision. Cette étape doit être achevée pour que la dégradation des performances liée à l'utilisation de l'arithmétique virgule fixe soit limitée. De nombreux systèmes embarqués de traitement du signal réalisent l'acquisition de données issues de capteurs. Ces données sont entachées d'un bruit dénommé bruit d'acquisition. Les performances du système sont fonctions du niveau de bruit d'acquisition ou des caractéristiques du signal acquis. Ainsi, la contrainte de précision sera déterminée en fonction de ce niveau de bruit d'acquisition et des caractéristiques du signal. De surcroît, il faut prendre en compte le rapport entre le niveau du signal et le niveau du bruit car il peut avoir une influence sur la dynamique des données au sein de l'application. Les applications de télécommunication sans fil possèdent ce type de propriété. Les données à l'entrée varient sur une dynamique très large. Ce phénomène est lié au canal de transmission qui est fluctuant. Ces différents éléments peuvent être exploités afin d'adapter au cours du temps la spécification virgule fixe en vue de réduire la consommation d'énergie.

Objectif de cette thèse

Notre travail de recherche a pour objectif d'optimiser la consommation d'énergie des systèmes numériques à travers les aspects arithmétiques. L'augmentation de la largeur des données permet d'améliorer la précision des calculs réalisés mais accroît la consommation d'énergie. Les expérimentations présentées dans [129] montrent le potentiel de gain en consommation d'énergie pouvant être obtenu en agissant sur la spécification virgule fixe. Entre deux contraintes de précision (90 dB et 30 dB), la consommation d'énergie d'un filtre adaptatif basé sur l'algorithme LMS est divisée par un facteur deux. Pour ce cas, la largeur des données passe de 21 bits à 11 bits. De plus, pour une même contrainte de précision, l'optimisation de la largeur des opérateurs permet d'obtenir un gain de 30% par rapport à une solution utilisant un codage des données standard avec une même largeur pour toutes les données.

Un premier objectif est de développer et mettre en œuvre une méthodologie d'optimisation de la spécification virgule fixe afin de minimiser la consommation d'énergie. Ce processus d'optimisation est réalisé lors de la phase de conception et de développement de l'application.

Un second objectif est de définir des techniques avancées permettant d'adapter la spécification virgule fixe de l'application au cours du temps en fonction de critères externes. En effet, la contrainte de précision des calculs peut évoluer au cours du temps en fonction d'éléments externes. Cette approche s'adresse plutôt aux systèmes autonomes en énergie utilisant des batteries. L'objectif est d'ajuster la consommation d'énergie liée aux traitements numériques afin de pouvoir préserver la charge des batteries.

Contributions

Ce travail de recherche s'articule autour de deux axes principaux. Dans un premier temps, le concept d'adaptation dynamique de la précision a été proposé [108, 110]. Celui-ci permet d'adapter la spécification virgule fixe en fonction des caractéristiques du signal d'entrée. Nous avons montré l'intérêt de notre approche d'adaptation dynamique pour réduire la consommation d'énergie. Pour l'énergie et pour la précision des calculs, des modèles analytiques ont été développés afin de pouvoir explorer l'espace de conception [109]. Ces modèles et ces développements ont permis de mettre en œuvre une chaîne complète de conversion en virgule fixe pour cette application. De plus un modèle de simulation a été développé afin de valider l'évaluation de performances. Dans un premier temps une contrainte de précision est déterminée en fonction du rapport signal à bruit en entrée du récepteur et ensuite la conversion virgule fixe est réalisée. Pour cela, la dynamique des données est déterminée en dissociant le signal utile du bruit. Ensuite, la consommation d'énergie du système est optimisée sous la contrainte de précision déterminée dans la première étape. En appliquant l'adaptation dynamique de la spécification virgule fixe sur un récepteur WCDMA, nous pouvons économiser jusqu'à 40% d'énergie dans le module de décodage des symboles (*rake receiver*) et jusqu'à 25% dans le module de recherche des trajets (*searcher*).

Dans un second temps, les algorithmes d'optimisation utilisés pour le processus de minimisation du coût de l'implantation sous contrainte de précision sont étudiés. Cette classe de problème est NP-difficile et le temps d'optimisation est extrêmement important pour résoudre des problèmes de grande taille. Nous proposons une amélioration des techniques

basées sur les algorithmes génétiques. Cette amélioration, en ajoutant une complexité de calcul négligeable, permet d'augmenter significativement la qualité de la solution obtenue. Ensuite, l'algorithme GRASP (*Greedy Randomized Adaptive Search Procedures*) est appliqué pour l'optimisation de la largeur des données [111]. Cet algorithme applique les tendances d'optimisation modernes en combinant un algorithme glouton aléatoire et un algorithme déterministe basé sur la recherche avec tabous. Cette approche permet de fournir de meilleurs résultats par rapport aux autres algorithmes dans toutes les applications examinées, avec une complexité de calcul raisonnable.

Publications

La liste des publications réalisées dans le cadre de ce travail de recherche est la suivante :

1. H.-N. Nguyen, D. Menard, R. Rocher, and O. Sentieys, "Energy reduction in wireless system by dynamic adaptation of the fixed-point specification," in *Proc. Conference on Design and Architectures for Signal and Image Processing (DASIP)*, Brussels, Belgium, Nov. 2008, pp. 132–139
2. H.-N. Nguyen, D. Menard, and O. Sentieys, "Dynamic precision scaling for low power WCDMA receiver," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Taipei, Taiwan, May 2009, pp. 205–208
3. H.-N. Nguyen, D. Menard, and O. Sentieys, "Design of optimized fixed-point WCDMA receiver," in *Proc. European Signal Processing Conference (EUSIPCO)*, Glasgow, UK, Aug. 2009, pp. 993–997
4. D. Menard, H.-N. Nguyen, F. Charot, S. Guyetant, J. Guillot, E. Raffin, and E. Casseau, "Exploiting reconfigurable SWP operators for multimedia applications," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 1717–1720
5. H.-N. Nguyen, D. Menard, and O. Sentieys, "Novel algorithms for word-length optimization," in *Proc. European Signal Processing Conference (EUSIPCO)*, Barcelona, Spain, Aug.–Sep. 2011, pp. 1944–1948
6. D. Menard, O. Sentieys, N. Herve, and H.-N. Nguyen, "High-level synthesis under fixed-point accuracy constraint," *Journal of Electrical and Computer Engineering*, pp. 1–26, Jan. 2012

Plan du document

Ce travail de recherche est présenté au travers de deux parties comprenant six chapitres. Le premier chapitre permet d'avoir une vue générale sur l'arithmétique virgule fixe ainsi que sur les méthodologies de conversion en virgule fixe. Tout d'abord, différents types de codage de données sont introduits. Ensuite, l'arithmétique virgule fixe est présentée. Finalement, une revue sur l'ensemble des méthodologies de conversion en virgule fixe et celle retenue au sein de l'équipe CAIRN est présentée.

Le premier axe de recherche se concentre sur l'optimisation dynamique de la précision en vue de réduire la consommation d'énergie. Cette partie comprend deux chapitres. Dans le chapitre 2, la méthode proposée est présentée. Tout d'abord, nous présentons les approches existantes d'adaptation de la précision. Ensuite, le principe de la méthode

proposée, est détaillé. Finalement, les supports d'exécution candidats pour ce type de technique sont présentés après avoir analysé les caractéristiques et la granularité en termes de largeur supportées, des plateformes existantes.

Le chapitre 3 est consacré à la présentation de l'application de notre approche sur différents systèmes de communication correspondant à une chaîne de transmission QPSK et aux systèmes CDMA et plus particulièrement un récepteur WCDMA utilisé pour les systèmes de télécommunication de troisième génération. La relation entre le taux d'erreurs binaires et la contrainte de précision du système a été établie. Ces études permettent de calculer analytiquement la dynamique de chaque opérateur et la précision du système en fonction du rapport signal sur bruit en entrée du récepteur de la chaîne de transmission.

La deuxième partie traite de l'optimisation de la largeur des données. Elle débute par un état de l'art présenté dans le chapitre 4. Ce chapitre introduit le problème d'optimisation combinatoire en général et présente les algorithmes utilisés dans l'optimisation des largeurs des opérateurs.

Dans le chapitre 5, les algorithmes génétiques (AG) sont considérés. Les avantages ainsi que les inconvénients des AG sont analysés et deux propositions permettant d'améliorer ces algorithmes sont présentées.

Le dernier chapitre de la deuxième partie propose une application des algorithmes de recherche locale stochastique pour l'optimisation des largeurs. Nous étudions les avantages et les inconvénients de l'algorithme GRASP et les propriétés permettant de résoudre les difficultés rencontrées par les autres algorithmes. Ce chapitre conclut avec les résultats montrant l'intérêt de GRASP par rapport aux algorithmes déterministes.

Finalement, nous concluons cette étude en résumant les apports de ce travail de recherche. Ensuite, différentes perspectives sont exposées, plus particulièrement sur la conception des opérateurs multi-précisions.

Chapitre 1

Conversion en virgule fixe

1.1 Introduction

L'objectif de ce chapitre est de présenter l'arithmétique virgule fixe et les méthodes existantes de conversion en virgule fixe. Dans la première partie, le problème de codage des données au sein d'une machine est abordé. Tout d'abord, différentes représentations des nombres sont présentées et ensuite les codages en virgule fixe et en virgule flottante sont analysés et comparés. Dans la seconde partie, le codage en virgule fixe est détaillé puis le processus de conversion en virgule fixe est explicité. Pour chaque transformation, les différentes approches disponibles au sein de la littérature sont brièvement présentées.

1.2 Codage des données

Dans cette partie, différentes représentations des nombres sont présentées. Tout d'abord, plusieurs systèmes de codage des nombres entiers sont exposés. Ensuite, basés sur ces systèmes, les codages des nombres réels en virgule fixe et en virgule flottante sont présentés.

1.2.1 Représentation des nombres entiers

Numération simple de position

Un entier p (compris entre 0 et $B^N - 1$) est décomposé en base B de façon unique selon l'expression suivante :

$$p = \sum_{i=0}^{N-1} a_i B^i, \tag{1.1}$$

avec a_i des chiffres compris entre 0 et $B-1$. La notation de cet entier est $(a_{N-1}...a_2a_1a_0)_B$ ou bien $(a_{N-1}...a_2a_1a_0)$ si la base B a été précisée auparavant.

Pour éviter l'ambiguïté entre le nombre codé et le codage, par la suite, nous dénommons Entier Simple Associé en base B (ESA_B), une chaîne $a_{N-1}...a_2a_1a_0$ de N chiffres compris

entre 0 et $B-1$ la quantité :

$$\mathrm{ESA}_B(a_{N-1}...a_2a_1a_0) = \sum_{i=0}^{N-1} a_i B^i. \tag{1.2}$$

Notation de type « signe – valeur absolue » (SVA)

Cette notation correspond à la présentation en numération simple de position de *la valeur absolue* en adjoignant un symbole présentant le *signe* du nombre à écrire. Ce système de numération possède un inconvénient majeur. Pour réaliser l'addition de deux nombres, un algorithme *d'addition* est utilisé si ces deux entiers sont de même signe et un algorithme *de soustraction* est utilisé dans le cas contraire.

Notation en complément à la base

Le système de numération des entiers, en *complément à la base B* permet d'éviter le problème de prise en compte du signe des opérandes présent pour le système SVA. Considérons une base B *paire*, la représentation de $p = (a_{N-1}...a_2a_1a_0)_{\bar{B}}$ est la suivante.

- Si $0 \leq p \leq B^N/2 - 1$, l'écriture est identique à celle de l'équation (1.1) :

$$\mathrm{ESA}_B(a_{N-1}...a_2a_1a_0) = p.$$

- Si $-B^N/2 \leq p \leq -1$, alors le codage est réalisé à l'aide de l'expression suivante

$$\mathrm{ESA}_B(a_{N-1}...a_2a_1a_0) = B^N + p.$$

Avec cette notation, un seul algorithme d'addition modulo B^N est nécessaire pour additionner tous les entiers compris entre $-B^N/2$ et $B^N/2-1$. Un dépassement de capacité est présent si et seulement si le signe du résultat obtenu pour l'addition de deux entiers de même signe est de signe différent de celui des opérandes. Si la base $B = 2$, cette notation devient complément à deux (CA2).

1.2.2 Codages des nombres réels

Codage en virgule fixe

Un réel x est écrit sous la forme $x = p.K$ où p est un entier codé par une des méthodes étudiées auparavant et K le facteur d'échelle. Ce facteur d'échelle est une puissance de la base, implicitement décidé avant : $K = B^{-n}$. Il permet de définir la position de la virgule : n est la distance entre la virgule et le bit de poids faible – *Least Significant Bit* (LSB). La figure 1.1 représente un nombre codé en virgule fixe avec un bit de signe s, m bits pour la partie entière et n bits pour la partie fractionnaire. Le nombre de bits total est $b = m + n + 1$.

La notation en virgule fixe est simple et permet d'effectuer les calculs rapidement. En contrepartie, comme le facteur d'échelle K est fixé, la notation en virgule fixe ne permet pas de représenter des nombres d'ordre de grandeur très différents. De plus, il n'est pas toujours facile de fixer a priori le facteur d'échelle si l'ordre de grandeur des valeurs prises par la donnée n'est pas connu. La notation en virgule flottante permet de résoudre ces problèmes.

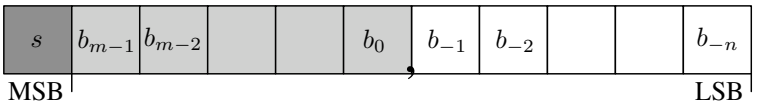


FIGURE 1.1: Codage en virgule fixe. Les paramètres m et n sont des entiers et représentent la distance, en nombre de bits, entre la virgule et respectivement les bits MSB, LSB. Le nombre de bits total est $b = m + n + 1$.

Codage en virgule flottante

Un réel x est écrit en virgule flottante, en base B avec M chiffres de mantisse et E chiffres d'exposant à l'aide du couple (m, e) tel que :

- m, appelé *mantisse* de x, est écrit en virgule fixe en base B sur M chiffres ;
- e, appelé *exposant* de x, est un entier écrit en base B sur E chiffres ;
- $x = m.B^e$

B^e joue le rôle du facteur d'échelle de la notation en virgule fixe. L'avantage par rapport à la virgule fixe est que ce facteur d'échelle s'adapte à la valeur à coder.

Il est à noter que la représentation d'un nombre n'est pas unique et les zéros à gauche de la mantisse nuisent à la précision. Par exemple, considérons la base 10 avec 4 chiffres de mantisse, π peut être représenté par $0.031 \cdot 10^2$ ou $3.142 \cdot 10^0$). Ces deux représentations ne possèdent pas la même précision. Afin de pallier ce problème, une représentation « sans zéros à gauche » est imposée et correspond à la représentation en *virgule flottante normalisée*. Dans ce cas, le premier chiffre de M est toujours 1 et ne figure pas dans la représentation.

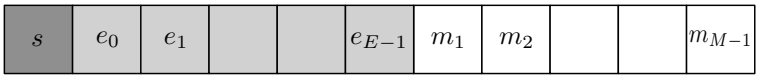


FIGURE 1.2: Codage en virgule flottante normalisée. Le premier chiffre m_0 est toujours égal à un et ne figure pas dans la représentation. s est le bit de signe.

La figure 1.2 représente le codage en virgule flottante normalisé avec deux parties : l'exposant e et la mantisse m, s est le bit de signe. La valeur de la mantisse m est calculée par

$$m = 1 + \sum_{k=1}^{M-1} m_k B^{-k}. \tag{1.3}$$

Plusieurs manières de représenter l'exposant sont disponibles. Lorsque la base est 2, la convention « exposant biaisé » est souvent utilisée. Dans ce cas, la valeur 2^{e-1} est ajoutée à l'exposant effectif afin de ne mémoriser que des exposants positifs, pouvant être plus facilement comparés.

Certaines représentations en virgule flottante avec différents nombres de bits sont définies dans la norme IEEE 754-2008 [75], dont quatre en base 2, avec des largeurs de 16, 32, 64 et 128 bits. La représentation *half precision* sur 16 bits a été introduite récemment et est utilisée dans les GPU (*Graphics Processing Unit*) comme nVidia GeForce FX, ILM OpenEXR.

Comparaison virgule fixe et virgule flottante

Généralement, la virgule flottante nécessite plus de bits pour la représentation des nombres. Comme l'exposant et la mantisse sont deux parties indépendantes, un certain nombre de bits pour chaque partie est nécessaire afin d'assurer la précision souhaitée et une dynamique suffisante pour supporter les différentes utilisations prévues. Les codages définis dans la norme IEEE 754-2008 et les plus utilisés sont au moins de 32 bits. Or, la majorité des données en virgule fixe est codée entre 6 et 32 bits. Il existe également des formats en virgule flottante utilisant moins de 32 bits pour certaines classes d'applications. Par exemple, la représentation *half precision* ou celle proposée dans [127], permettent de coder en virgule flottante sur 16 bits des réels entre 0 et 1.

Étant codée sur un nombre plus important de bits, la représentation en virgule flottante permet d'obtenir une meilleure précision par rapport à celle en virgule fixe. La présence d'un facteur d'échelle variable permet de coder aussi précisément les petits et les grands nombres. L'arithmétique virgule flottante montre aussi son intérêt par sa dynamique très importante. La virgule flottante se trouve donc dans quasiment toutes les architectures d'ordinateur actuelles et est utilisée pour les calculs scientifiques.

Néanmoins, le codage des données sur un nombre de bits plus important et l'utilisation d'opérateurs plus complexes pour pouvoir traiter la mantisse et l'exposant, conduit à des architectures ayant un coût plus élevé en termes de surface et de consommation d'énergie. Ainsi, l'implantation efficace des applications de traitement numérique du signal dans les systèmes embarqués privilégie l'utilisation de l'arithmétique virgule fixe. Ainsi, la grande majorité des applications DSP embarqués est mise en œuvre sur des architectures en virgule fixe [36, 38, 141, 39].

1.3 Arithmétique virgule fixe

Dans la suite de ce document, nous utilisons la notation (b, m, n) pour le format d'un nombre codé sur b bits représenté à la figure 1.1. m est donc le nombre de bits consacrés à la partie entière tandis que n est le nombre de bits pour la partie fractionnaire. En présence d'un bit de signe, la relation $b = m + n + 1$ est obtenue. De plus, sans autre indication, la représentation par défaut sera en CA2.

1.3.1 Opérations arithmétiques en virgule fixe

Le domaine de définition (dynamique) d'un entier a au format (b, m, n) avec un bit de signe et une représentation en CA2 est

$$\mathcal{D}_{\mathcal{D}} = [-2^m, 2^m - 2^{-n}] \tag{1.4}$$

Dans un format non-signé ($b = m + n$), la dynamique est $\mathcal{D}_{\mathcal{D}} = [0, 2^m - 2^{-n}]$.

Addition

Pour une addition $c = a + b$ en virgule fixe, il est nécessaire que les deux opérandes a et b possèdent un format commun : le type de représentation, la largeur de la partie entière,

la largeur de la partie fractionnaire doivent être identiques. Dans le cas d'une conversion sans perte d'information, le format commun minimal (b_c, m_c, n_c) est le suivant :

$$\begin{aligned} m_c &= \max(m_a, m_b) \\ n_c &= \max(n_a, n_b) \\ b_c &= m_c + n_c + 1. \end{aligned} \tag{1.5}$$

L'extension d'un format (b_a, m_a, n_a) à un format (b_c, m_c, n_c) suit les règles suivantes :
- le bit de signe reste le même ;
- pour la partie entière, les $m_c - m_a$ bits supplémentaires prennent la valeur du bit de signe et sont ajoutés à gauche ;
- pour la partie fractionnaire, les $n_c - n_a$ bits supplémentaires prennent la valeur 0 et sont ajoutés à droite.

Le format du résultat est présenté à l'équation (1.6). La partie entière nécessite un bit supplémentaire si le résultat n'appartient pas au domaine de définition $\mathcal{D}_c = [-2^{m_c}, 2^{m_c} - 2^{-n_c}]$, sinon un débordement est présent.

$$\begin{aligned} m_{\text{Add}} &= \begin{cases} m_c + 1 & \text{si } a + b \notin \mathcal{D}_c \\ m_c & \text{si } a + b \in \mathcal{D}_c \end{cases} \\ n_{\text{Add}} &= n_c \end{aligned} \tag{1.6}$$

Multiplication

Dans le cas général, les deux opérandes peuvent avoir des largeurs différentes. Le nombre de bits du résultat est la somme du nombre de bits des opérandes, avec un bit de signe. La relation entre le nombre de bits du résultat et ceux des opérandes est la suivante :

$$\begin{aligned} m_{\text{Mult}} &= m_a + m_b + 1 \\ n_{\text{Mult}} &= n_a + n_b \\ b_{\text{Mult}} &= b_a + b_b. \end{aligned} \tag{1.7}$$

1.3.2 Règles de codage en virgule fixe

Nous présentons dans ce paragraphe le processus de codage d'une donnée arbitraire en virgule fixe. Les différentes lois de quantification et de dépassement sont détaillées. Ces caractéristiques sont utiles pour l'étude statistique du bruit de quantification.

Soit x une valeur appartenant au domaine $\mathcal{D}$ et y une valeur du domaine de définition $\mathcal{D}_\mathcal{R}$ de codage choisi. Le domaine $\mathcal{D}_\mathcal{R}$ est borné par les valeurs $X_{\min}$ et $X_{\max}$. Nous définissons le sous-ensemble $\mathcal{D}_\mathcal{D}$ de $\mathcal{D}$ regroupant l'ensemble des valeurs de $\mathcal{D}$ comprises dans l'intervalle $[X_{\min}; X_{\max}]$. Le processus de quantification correspond à l'opération de réduction d'une valeur arbitraire x à une valeur représentable y. Ce processus est régi par deux lois présentées ci-dessous.

La loi de dépassement permet d'associer à une valeur x de $\mathcal{D}$ une valeur de $\mathcal{D}_\mathcal{D}$. Elle définit plus précisément le comportement pour les valeurs présentes en dehors du domaine

$\mathcal{D}_\mathcal{D}$. Nous associons à cette loi une fonction de dépassement définie ci-dessous :

$$f_D(x) = \begin{cases} x & \forall x \in \mathcal{D}_\mathcal{D} \\ D(x) & \forall x \notin \mathcal{D}_\mathcal{D} \end{cases} . \tag{1.8}$$

La loi de quantification définit les valeurs représentables y à associer à l'ensemble des valeurs x appartenant au domaine $\mathcal{D}_\mathcal{D}$. La fonction de quantification associée est la suivante :

$$f_Q(x) = Q(x) \quad x \in \mathcal{D}_\mathcal{D}. \tag{1.9}$$

Le processus de quantification global peut s'exprimer sous la forme suivante :

$$x \mapsto y = f_Q(f_D(x)) = \begin{cases} Q(x) & \forall x \in \mathcal{D}_\mathcal{D} \\ D(x) & \forall x \notin \mathcal{D}_\mathcal{D} \end{cases} . \tag{1.10}$$

Loi de dépassement

Arithmétique de saturation Cette loi de saturation consiste à choisir la valeur du domaine $\mathcal{D}_\mathcal{D}$ la plus proche de la valeur à représenter x :

$$D(x) = \begin{cases} X_{\min} & \forall x < X_{\min} \\ X_{\max} & \forall x > X_{\max} \end{cases} \tag{1.11}$$

Arithmétique modulaire Cette loi remplace x par une valeur congrue à x modulo $X_{\max} - X_{\min}$ et appartenant au domaine $\mathcal{D}_\mathcal{D}$.

Loi de quantification

Le domaine représentable $\mathcal{D}_\mathcal{R}$ est composé de N valeurs y_i avec $i = 1, 2, ..., N$ et le sous-domaine $\mathcal{D}_\mathcal{D}$ est subdivisé en N intervalles juxtaposés $\Delta_i = [u_i; u_{i+1}]$. Une loi de quantification associe à tout x appartenant au domaine Δ_i la valeur y_i :

$$Q(x) = y_i \quad \forall x \in \Delta_i. \tag{1.12}$$

Loi de quantification par arrondi La loi de quantification par arrondi consiste à choisir la valeur représentable la plus proche de la valeur à quantifier en prenant la médiane de chaque intervalle :

$$y_i = \frac{u_i + u_{i+1}}{2} = u_i + \frac{q}{2}, \tag{1.13}$$

avec $q = u_{i+1} - u_i$ appelé le pas de quantification. Les intervalles Δ_i possèdent la même largeur.

Loi de quantification par troncature Cette loi de quantification consiste à tronquer un certain nombre de bits de poids faible. Dans le cas d'une représentation en complément à 2, la loi revient à prendre la valeur représentable immédiatement inférieure à la valeur à quantifier :

$$y_i = u_i. \tag{1.14}$$

1.3.3 Implantation

L'utilisation native de la virgule flottante requiert une unité de calcul en virgule flottante (en anglais *Floating Point Unit*, soit *FPU*) qui n'est disponible que sur des processeurs haut de gamme. En revanche, le calcul en virgule fixe est disponible pour tous les processeurs, que ce soit un DSP, un SoC ou n'importe quel processeur pour des systèmes embarqués. Les mêmes opérateurs sont utilisés dans le cas de représentation des nombres en entier et des nombres en virgule fixe. La seule différence réside dans l'interprétation des données.

Pour des simulations en virgule fixe, les langages de programmation de haut niveau sont utilisés. Ces langages sont indépendants de l'architecture et la plupart d'entre eux ne supporte pas de donnée en virgule fixe. Différentes techniques d'émulation de la virgule fixe sont utilisées. Soit après chaque opération, le résultat est converti en un format équivalent à la virgule fixe. Cette étape peut être réalisée directement avec des opérateurs surchargés (gFix [80], types `ac_fixed`, types `sc_fixed` de SystemC, pour le langage C/C++). Dans MATLAB, la boîte à outils *Fixed-Point* définit de nouveaux types virgule fixe complets et disponibles également dans Simulink. D'autres outils utilisant des caractéristiques de la machine hôte pour accélérer la simulation en virgule fixe sont disponibles (pFix [80], FRIDGE [79], ...). Le sous ensemble *Embedded MATLAB* permet d'accélérer nettement la simulation en virgule fixe par apport à une simulation en virgule fixe non accélérée. Des logiciels de conception de système sont équipés aussi d'outils de simulation en virgule fixe, par exemple SPW (CoWare-Synopsys) ou CoCentric (Synopsys).

1.4 Méthodologie de conversion en virgule fixe

La conversion d'un algorithme spécifié en virgule flottante vers une représentation en virgule fixe peut être divisée en deux étapes. La première étape consiste à déterminer la largeur m_i de la partie entière de chaque donnée. Cette largeur (le nombre de bits) doit permettre la représentation de toutes les valeurs possibles des données. Tout d'abord, le domaine de définition de chaque donnée est évalué. Ensuite, ce résultat est utilisé pour déterminer la position de la virgule qui minimise, pour chaque donnée, la largeur de la partie entière et qui évite le débordement. De plus, des opérations de recadrage sont insérées afin d'aligner le format virgule fixe au domaine de définition ou aux autres données.

La deuxième étape consiste à déterminer le nombre de bits pour la partie fractionnaire. Le nombre de bits n_i définit la précision des calculs. L'objectif est d'optimiser la largeur des données. Le coût de l'implantation est minimisé sous contrainte de précision.

1.4.1 Évaluation de la dynamique

La première étape dans la conversion en virgule fixe correspond à l'estimation de la dynamique, ou du domaine de définition, de chaque donnée. Cette valeur permet de déterminer la position de la virgule par rapport au bit le plus significatif. Cette position est définie de manière à garantir l'absence de débordement sur chaque donnée où à contrôler la probabilité de débordement. Ainsi, il est nécessaire d'utiliser des méthodes précises.

Méthodes basées sur l'estimation de la dynamique

Les méthodes présentées dans cette section ont pour but de déterminer la dynamique en vue d'éviter la présence de débordement. Les approches analytiques permettent de garantir l'absence de débordement.

Approches analytiques La première méthode correspond à l'arithmétique d'intervalles [78]. Les intervalles de définition $[\underline{x}, \overline{x}]$ et $[\underline{y}, \overline{y}]$ des données x et y en entrée sont propagés à travers le système selon les relations suivantes

$$z = x + y : \qquad [\underline{z}, \overline{z}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \tag{1.15}$$

$$z = x - y : \qquad [\underline{z}, \overline{z}] = [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \tag{1.16}$$

$$z = xy : \qquad [\underline{z}, \overline{z}] = [\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})] \tag{1.17}$$

Cette méthode garantit l'absence de débordement mais l'estimation est pessimiste. Par exemple pour $x \in [-1, 1]$, l'opération $x - x$ donne $[-2, 2]$ au lieu de 0 parce que la corrélation entre les données n'est pas prise en compte.

Afin de résoudre ce problème, l'arithmétique affine a été introduite [41, 40]. Une variable x est développée à l'aide de sa représentation affine selon

$$x = x_0 + x_1\epsilon_1 + ... + x_N\epsilon_N, \tag{1.18}$$

où $\epsilon_i \in [-1, 1]$ et x_i sont des constantes. Soit y une autre variable avec la représentation affine, alors

$$y = y_0 + y_1\epsilon_1 + ... + y_N\epsilon_N. \tag{1.19}$$

L'expression affine de chaque entrée est propagée au sein du système. Une règle de propagation est associée à chaque opération avec l'objectif que la sortie soit représentée par une expression affine. Les expressions pour l'addition et la soustraction sont respectivement

$$x + y = x_0 + y_0 + \sum_{i=1}^{N}(x_i + y_i)\epsilon_i, \tag{1.20}$$

$$x - y = x_0 - y_0 + \sum_{i=1}^{N}(x_i - y_i)\epsilon_i. \tag{1.21}$$

Pour la multiplication une approximation doit être réalisée afin d'obtenir une expression affine telle que

$$xy = x_0y_0 + \sum_{i=1}^{N}(y_0x_i + x_0y_i)\epsilon_i + z_k\epsilon_k, \tag{1.22}$$

$$\text{avec} \qquad z_k = \sum_{i=1}^{N}|x_i| \sum_{i=1}^{N}|y_i|.$$

Cette approche permet de résoudre la corrélation spatiale parce que $x - x$ est égal à 0. Cependant, la corrélation temporelle n'est pas prise en compte. Ce problème est particulièrement gênant pour les applications de traitement numérique du signal dans lesquelles les signaux sont considérés avec différents retards.

Norme L1 Dans le cas des systèmes linéaires invariant dans le temps (LTI) [1], la notion de réponse impulsionnelle peut être utilisée afin d'avoir une expression de la sortie en fonction uniquement des entrées et la norme L1 peut être utilisée [99]. Considérons un système linéaire ayant N_e entrées x_i. Soit y_k une donnée du système définie par

$$y_k(n) = \sum_{i=0}^{N_e-1} h_{ik}(n) * x_i(n), \tag{1.23}$$

avec $h_{ik}(n)$ la réponse impulsionnelle entre $x_i(n)$ et $y_k(n)$.

La valeur absolue de la donnée y_k est égale à

$$|y_k(n)| = \left| \sum_{i=0}^{N_e-1} \sum_{m=-\infty}^{+\infty} h_{ik}(m) x_i(n-m) \right|. \tag{1.24}$$

La valeur maximale $y_{k-\max}$ de y_k est obtenue par l'inégalité triangulaire et est égale à

$$\max_n (|y_k(n)|) \leq \sum_{i=0}^{N_e-1} \max_n (|x_i(n)|) \sum_{m=-\infty}^{+\infty} |h_{ik}(m)| = y_{k-\max}. \tag{1.25}$$

L'équation (1.25) permet de déterminer la valeur maximale de chaque donnée du système, à condition que la réponse impulsionnelle de la fonction de transfert entre cette donnée et chaque entrée du système soit connue.

Approches statistiques Quelques méthodes sont basées sur la simulation de Monte-Carlo pour estimer la dynamique des données [80, 144, 81]. Dans [81], la moyenne $\mu(x)$ et l'écart-type $\sigma(x)$ de la donnée x sont déterminés à l'aide des signaux obtenus en simulation. Ensuite, la dynamique $R(x)$ est déterminée par

$$R(x) = |\mu(x) + n\sigma(x)|, \tag{1.26}$$

avec n choisi par l'utilisateur (généralement entre 4 et 16).

Cependant, cette méthode n'est appropriée que si la distribution de x suit une loi unimodale et symétrique. Dans [80], une méthode plus élaborée est proposée où la symétrie et la moyenne de chaque donnée sont prises en compte.

Méthodes basées sur la probabilité de débordement

Les méthodes présentées dans cette section ont pour but de déterminer la dynamique pour une certaine probabilité de débordement.

1. Un système LTI est un système linéaire (la sortie est une fonction linéaire de l'entrée) dans lequel la relation entre l'entrée et sortie est identique au cours du temps.

Méthodes stochastiques Ces méthodes permettent de modéliser le signal sous la forme d'un processus stochastique afin de déterminer sa dynamique [151]. Les signaux d'entrée sont modélisés sous la forme d'une somme pondérée de variables aléatoires. Ensuite, les paramètres associés au modèle de chaque entrée sont propagés dans le système afin d'obtenir les paramètres de la sortie considérée. À partir de la modélisation de la sortie, la fonction de densité de probabilité de cette sortie est déterminée et la dynamique obtenue pour une probabilité de débordement donnée en est déduite.

Pour les systèmes linéaires, la méthode de *Karhunen-Loève Expansion* (KLE) peut être utilisée. Une variable aléatoire $p(t)$ centrée peut se décomposer selon

$$p(t) = \sum_{i=0}^{\infty} \sqrt{\lambda_i} f_i(t) \mu_i \tag{1.27}$$

où $f_i(t)$ sont les fonctions propres, λ_i les valeurs propres de la fonction de covariance R, et μ_i des variables aléatoires de variance unitaire, de moyenne nulle et non-corrélées entre elles.

Pour un système non-linéaire, la méthode *Polynomial Chaos Expansion* (PCE) est utilisée. Un processus aléatoire de second ordre $X(\Theta)$ est décomposé selon

$$X(\Theta) = \sum_{i=0}^{\infty} a_i \Psi_i(\zeta(\Theta)) \tag{1.28}$$

où a_i sont des coefficients, $\zeta(\Theta)$ une variable aléatoire multi-dimensionnelle et Ψ_i les polynômes d'Hermite ou polynômes orthogonaux.

Théorie des valeurs extrêmes La théorie des valeurs extrêmes [56] permet de modéliser la distribution des valeurs minimales et maximales d'un processus. D'après [82], les maxima et les minima d'un processus aléatoire suivent la distribution de Gumbel

$$f(x) = \frac{1}{\beta} e^{\frac{-(x-\mu)}{\beta}} e^{-e^{\frac{-(x-\mu)}{\beta}}} \tag{1.29}$$

$$\beta = \frac{\sigma_x \sqrt{6}}{\pi} \tag{1.30}$$

$$\mu = \mu_x - \beta\gamma \tag{1.31}$$

où μ_x et σ_x sont respectivement la moyenne et l'écart-type de x et γ la constante d'Euler ($\approx 0{,}5772$).

N tests sont exécutés et les N minima et maxima sont extraits de ces simulations et leur valeur moyenne μ_x et l'écart-type σ_x peuvent être calculés. Soit P la probabilité de débordement, la valeur maximale x_{max} est déduite de l'équation (1.29) et conduit à [116]

$$x_{max} = \mu - \sigma_x \ln(\ln(\frac{1}{P})). \tag{1.32}$$

L'expression (1.32) permet de déterminer le domaine de définition de la donnée avec une certaine probabilité de débordement.

Conclusions

Les méthodes présentées peuvent être divisées en deux groupes correspondant aux méthodes analytiques et aux méthodes basées sur la simulation. L'intérêt des méthodes basées sur la simulation est l'adaptabilité à tout système en utilisant les informations liées à la statistique des données. Ces méthodes ne sont cependant appropriées que pour les signaux d'entrée proches de ceux utilisés pour la simulation. La fiabilité de ces méthodes dépend de la connaissance statistique du signal et/ou du nombre de simulations. Les méthodes analytiques, bien que pessimistes, garantissent l'absence de débordement.

1.4.2 Optimisation de la largeur des données

La seconde étape du processus de conversion en virgule fixe correspond à la détermination du nombre de bits pour la partie fractionnaire. Le nombre de bits choisi pour la partie fractionnaire influence la précision des calculs mais aussi le coût de l'implantation. Ainsi, l'objectif est de minimiser le coût de l'implantation tant que les contraintes de performance de l'application sont respectées. L'évaluation directe des performances étant assez complexe, une métrique intermédiaire permettant de quantifier la précision des calculs est utilisée. Le processus d'optimisation nécessite d'évaluer le coût de l'implantation et la précision des calculs

Évaluation de la précision

L'utilisation de l'arithmétique virgule fixe limite la précision des calculs et conduit à une erreur de quantification en sortie correspondant à la différence entre le résultat en précision finie et celui en précision infinie. Cette erreur de quantification est considérée comme un bruit ajouté au résultat. Il est donc nécessaire de vérifier que le comportement de l'algorithme utilisant l'arithmétique virgule fixe change dans une limite raisonnable. Pour cela, une métrique de précision doit être déterminée et cette métrique sera calculée en fonction des largeurs de données.

Il existe différentes métriques d'évaluation de la précision, comme les bornes du bruit, les moments du bruit ou le nombre de bits significatifs associés à chaque donnée. La métrique la plus utilisée dans le traitement numérique du signal est la puissance du bruit de quantification, ou le rapport signal à bruit de quantification (RSBQ). Soit P_b la puissance du bruit de quantification et P_y la puissance du signal en sortie, le RSBQ est défini par

$$\mathrm{RSBQ} = \frac{P_b}{P_y}, \tag{1.33}$$

ou en dB

$$\mathrm{RSBQ}_{\mathrm{dB}} = 10 \log \frac{P_b}{P_y}. \tag{1.34}$$

Dans cette section, les méthodes d'évaluation du RSBQ en fonction des largeurs de données sont présentées. Ces méthodes peuvent à nouveau être classées en deux familles : les méthodes statistiques et les méthodes analytiques.

Méthodes statistiques L'évaluation des performances d'un système en virgule fixe peut être réalisée à partir de simulations en virgule fixe. Soit y le résultat en sortie de la simulation en virgule flottante et y_{fixed} le résultat de la simulation en virgule fixe, la puissance du bruit de quantification est considérée égale à

$$P_b = E\left[(y - y_{\text{fixed}})^2)\right], \tag{1.35}$$

sous réserve d'un nombre de simulations suffisant pour la méthode de Monte–Carlo. La simulation en virgule flottante est considérée comme très proche du résultat en précision infinie car l'erreur de quantification associée à cette représentation est très faible par rapport à celle associée à l'arithmétique en virgule fixe.

Il existe différentes techniques de simulation en virgule fixe. La technique la plus simple consiste à convertir après chaque opération en virgule flottante le résultat en virgule fixe par arrondi ou par troncature. Dans [80], les auteurs proposent d'utiliser la *surcharge* des opérateurs pour implémenter les opérations en virgule fixe à travers la classe gFix. Dans MATLAB, la boîte à outils *Fixed-Point* fournit des types en virgule fixe assez complets. D'autres outils utilisent les caractéristiques de la machine hôte pour accélérer la simulation en virgule fixe (pFix [80], FRIDGE [79], etc.).

Cependant, le temps de simulation reste relativement important. Pour un seul changement dans la spécification virgule fixe, une nouvelle simulation est requise. De plus, pour avoir les informations statistiques de l'erreur de quantification, comme la valeur moyenne de la puissance, il est nécessaire de réaliser des simulations sur un nombre de points assez élevé. Afin de réduire le temps de simulation, les méthodes analytiques peuvent être utilisées.

Méthodes analytiques L'objectif des méthodes analytiques est de définir l'expression mathématique de la métrique de précision en fonction des largeurs de données. L'obtention de cette expression mathématique peut nécessiter un temps d'exécution relativement important mais cette expression est déterminée une seule fois. Ensuite, chaque évaluation de la précision correspond à l'évaluation d'une fonction mathématique dans laquelle le résultat est obtenu très rapidement.

Pour calculer l'expression analytique de la puissance du bruit de quantification, les approches existantes sont basées sur la théorie de la perturbation. Les valeurs en précision finie correspondent aux valeurs en précision infinie entachées d'une perturbation dont l'amplitude est très faible par rapport à l'amplitude des valeurs en précision infinie. Cette perturbation peut être assimilée à un bruit. Chaque source de bruit b_i se propage au sein du système et contribue au bruit global en sortie du système. Cette propagation doit être modélisée pour obtenir l'expression du bruit de quantification en sortie [130]. Un développement en série de Taylor du premier ordre est utilisé pour linéariser le comportement des opérations en présence d'un bruit de quantification en entrée.

Le bruit de sortie b_y est la somme de toutes les contributions des N_e sources de bruit. Le moment du second ordre de b_y peut être exprimé comme une somme pondérée des paramètres statistiques des sources de bruit, ce qui donne [97]

$$E\left[b_y^2\right] = \sum_{i=1}^{N_e} K_i \cdot \sigma_{b_i}^2 + \sum_{i=1}^{N_e}\sum_{j=1}^{N_e} L_{ij} \cdot \mu_{b_i}\mu_{b_j}. \tag{1.36}$$

Les termes μ_{b_i} et $\sigma^2_{b_i}$ sont respectivement la moyenne et la variance de la source de bruit b_i. Les termes K_i et L_{ij} sont des constantes et dépendent du système situé entre b_i et la sortie. Ainsi, ces termes sont déterminés une seule fois pour l'obtention de l'expression analytique de la précision.

Conclusions Les deux types d'approche permettant d'évaluer la précision d'une implémentation en virgule fixe ont été résumés. Les méthodes statistiques basées sur la simulation de l'algorithme en virgule fixe sont plus simples et applicables à tous les types d'algorithme de traitement numérique du signal. Cependant, ces techniques conduisent à des temps de simulation élevés et plus particulièrement si elles sont intégrées au sein d'un processus d'optimisation du format des données où un grand nombre de simulations est nécessaire. En effet, le surcoût dans l'émulation de la virgule fixe augmente le temps de simulation par rapport à la virgule flottante.

La seconde approche correspond aux méthodes analytiques permettant d'obtenir l'expression de la métrique de précision. Une expression fournissant des informations sur le comportement du bruit au sein de l'algorithme en virgule fixe est calculée analytiquement une seule fois. Ensuite, le temps d'évaluation de la précision d'un spécification virgule fixe correspond au temps d'évaluation de cette expression, ce qui est définitivement plus faible que le temps de simulation de l'application en virgule fixe. Le gain en temps des méthodes analytiques est très important sauf lorsque le nombre de simulations est faible.

Optimisation des largeurs

Soit $\boldsymbol{b}$ une spécification virgule fixe, c'est-à-dire une combinaison de la largeur des opérations. $\boldsymbol{b}$ est un vecteur de N éléments correspondant aux largeurs des opérations :

$$\boldsymbol{b} = [b_1, b_2, ..., b_N] \tag{1.37}$$

où N est le nombre d'opérations, b_k est la largeur (le nombre de bits) pour l'opération k. Le nombre de bits pour la partie entière a été déterminé dans l'étape précédente et ne varie pas pendant la procédure d'optimisation. Soit λ la métrique de précision et $\mathcal{C}$ le coût correspondant à cette spécification. L'objectif est de minimiser le coût en conservant une précision supérieure à un seuil $\lambda_{\min}$:

$$\min \mathcal{C}(\boldsymbol{b}) \quad \text{avec} \quad \lambda(\boldsymbol{b}) \geq \lambda_{\min}. \tag{1.38}$$

Ceci est un problème d'optimisation combinatoire. La taille N du problème peut être réduite en regroupant des opérations [68] de manière à ce que les opérations d'un même groupe possèdent le même format. Ainsi, ce regroupement nécessite une connaissance préalable de la largeur des opérateurs. Donc, pour un meilleur résultat, la synthèse d'architecture doit être couplée avec l'optimisation des largeurs.

Un problème d'optimisation combinatoire est NP–difficile et le temps nécessaire pour trouver le résultat est trop important lorsque la taille du problème est non trivial. Plusieurs approches sont proposées pour approximer ce résultat. Une étude des méthodes d'optimisation des largeurs ainsi que de nouveaux algorithmes font l'objet de la deuxième partie de cette thèse.

1.4.3 Conclusions

Dans la dernière partie de ce chapitre, nous avons présenté la méthodologie de conversion en virgule fixe d'un algorithme de traitement du signal. Les deux étapes correspondant à la détermination de la dynamique et à l'optimisation de la largeur des données ont été détaillées. Pour la première étape, plusieurs approches sont disponibles. Les approches analytiques basées sur l'arithmétique d'intervalle ou affine conduisent à des valeurs assez pessimistes. Les propositions récentes d'approches fixant la probabilité de débordement permettent d'améliorer l'estimation de l'intervalle de chaque donnée. Pour la deuxième étape, les approches analytiques sont plus avantageuses en termes de temps de calcul. Ces techniques sont viables si des estimateurs précis sont mis en œuvre pour un nombre important de systèmes de traitement du signal.

Première partie

Adaptation dynamique des largeurs

L'objectif de cette première partie est d'optimiser dynamiquement la spécification virgule fixe afin de l'adapter aux contraintes de qualité, de service ou de nature du signal d'entrée. La contrainte de précision peut être modifiée au cours du temps en fonction des performances souhaitées en sortie de l'application et de la qualité de service désirée. Différentes configurations, correspondant chacune à une spécification virgule fixe particulière, sont disponibles au sein du système embarqué. L'adaptation de la configuration à l'évolution de la contrainte de précision va permettre de réduire la consommation d'énergie et ainsi d'augmenter l'autonomie des systèmes embarqués sur batterie.

Pour de nombreux systèmes de traitement du signal, leurs performances dépendent du rapport signal à bruit (RSB) à sa sortie. Ce dernier est directement lié à la qualité du signal en entrée et ainsi au RSB en entrée. La contrainte de précision est définie telle que le bruit de quantification lié au codage en virgule fixe soit inférieur au bruit dans le système afin que la dégradation des performances liée au passage en virgule fixe soit limitée. En conséquence, la contrainte de précision va varier en fonction de la qualité du signal en entrée du système et la spécification virgule fixe peut être adaptée en fonction du signal d'entrée.

Dans le premier cas, l'évolution de la spécification virgule fixe est à l'initiative de l'utilisateur ou du système global de contrôle de l'application. Dans le second cas, les changements de la spécification virgule fixe doivent être décidés en fonction de la qualité du signal d'entrée. Ainsi, les techniques permettant de mesurer cette qualité devront être étudiées et mises en œuvre. De plus, les stratégies de modification de la spécification virgule fixe devront être définies. Les gains en termes de consommation d'énergie, apportés par notre approche seront mesurés sur plusieurs applications. Cette technique, dénommé *Dynamic Precision Scaling* (DPS) ou *adaption dynamique de la précision* (ADP), présentée à la figure 2.1, sera détaillée dans cette partie.

Chapitre 2

Méthodologie d'adaptation dynamique de la précision

2.1 Introduction

Le dimensionnement de la largeur des données d'un système en virgule fixe influence la consommation d'énergie de celui-ci. La puissance consommée au sein d'un circuit VLSI correspond à la somme de la puissance statique et dynamique. La puissance dynamique est liée au taux d'activité du circuit, à la tension d'alimentation, à la fréquence d'horloge et à la capacité équivalente du circuit. Cette dernière dépend de la technologie utilisée. La réduction de la largeur des données permet de diminuer la largeur des bus, des unités fonctionnelles et de la mémoire et ainsi, diminuer le taux d'activité du circuit et en conséquence la puissance consommée. La puissance statique est liée aux courants de fuite dans les transistors et dépend du nombre de transistors présents dans le circuit. La réduction de la largeur des données va permettre de diminuer le nombre de transistors présents au sein du circuit et ainsi diminuer la puissance statique. La réduction de la largeur des données en vue de réduire la consommation d'énergie s'accompagne d'une augmentation de l'erreur entre la valeur réelle et la valeur codée. Cette erreur de quantification diminue les performances de l'application. Les expérimentations réalisées dans [129] permettent d'illustrer ce compromis entre le coût de l'implantation et la précision des calculs. Un filtre adaptatif de type moindres carrés (LMS pour *Least Mean Squares*) a été étudié. La consommation d'énergie est divisée par un facteur deux entre les deux spécifications de virgule fixe pour un rapport signal sur bruit de quantification (RSBQ) de 90 dB et 30 dB.

L'approche traditionnelle de conception d'un système en virgule fixe est basée sur le principe du pire cas. Pour un récepteur de communication numérique, le RSB maximal, la dynamique maximale d'entrée et les conditions de canal les plus bruitées sont considérés. Néanmoins, le bruit et le niveau des signaux évoluent au cours du temps. De plus, le débit des données dépend du service (vidéo, image, parole). La performance requise (taux d'erreurs binaires) est liée aux services utilisés. Ces différents éléments montrent que la spécification virgule fixe dépend d'éléments extérieurs comme le niveau de bruit, la dynamique du signal d'entrée ou la qualité de service souhaitée. Cette spécification peut donc être adaptée au cours du temps pour réduire la consommation d'énergie moyenne. Dans

la suite, ces éléments extérieurs sont représentés par la métrique p.

Dans la section 2.2, les méthodes existantes d'adaptation au cours du temps de la spécification virgule fixe sont présentées. Dans la section 2.3, le principe d'adaptation dynamique de la précision proposé est défini. Cette technique nécessite un support d'exécution particulier défini dans la section 2.4. Après une présentation des alternatives disponibles pour l'implantation d'applications de traitement du signal et de l'image, la granularité, en termes de largeur de données, des supports d'exécution existants est analysée.

2.2 Méthodes d'adaptation existantes

2.2.1 Adaptation en fonction de la qualité de service désirée

Dans [152, 118], une adaptation de la spécification virgule fixe en fonction de la qualité de service souhaitée en sortie du système est proposée. Quatre niveaux de qualité de service sont considérés et à chaque niveau est associé une spécification virgule fixe. L'application ciblée est une transformée en cosinus discret à deux dimensions (DCT 2D). Cette DCT 2D permet de compacter l'énergie sur les composantes basses fréquences afin de compresser l'information dans le domaine de l'audio et de la vidéo. Cette propriété est aussi utilisée pour adapter la largeur des données en fonction de la qualité de service souhaitée. En fonction de cette qualité souhaitée, plus ou moins de précision est affectée au calcul des composantes hautes fréquences dont l'influence sur le résultat global est plus limitée. La technique proposée est très liée aux propriétés associées à l'application considérée et il n'est pas défini de méthode générale pour adapter la spécification virgule fixe en fonction de la qualité de service souhaitée.

2.2.2 Adaptation en fonction du mode de fonctionnement

Dans [112], différents compromis entre la précision et l'énergie sont explorés dans le cadre de la radio logicielle (SDR, pour *Software Defined Radio*). Certaines normes, telle la norme IEEE 802.11n (réseau local sans fil), offrent de multiples configurations en fonction des conditions de canal et du débit souhaité. Différents modes (schéma de modulation et taux de codage) sont proposés. Pour chaque mode, une spécification virgule fixe optimisée est déterminée et conduit à une implémentation spécifique. Le choix d'une spécification virgule fixe pour chaque schéma de modulation et taux de codage diminue la consommation moyenne d'énergie par un facteur trois dans le scénario considéré. Dans cette approche, l'adaptation de la spécification virgule fixe est uniquement liée au schéma de modulation et au taux de codage.

2.2.3 Adaptation en fonction des performances de l'application

Dans [155], une architecture VLSI avec une largeur réglable pour un démodulateur OFDM (*Orthogonal Frequency Division Multiplexing*) est proposée. La largeur est déterminée au moment de l'exécution en fonction de l'erreur observée en sortie du système. Pour cela, des symboles de *recherche de largeur* sont insérés dans la trame. Cette approche permet d'économiser 32 % et 24 % de la consommation d'énergie pour différents canaux de transmission. Cette technique nécessite un matériel spécifique et de l'énergie est gaspillée

pour le processus d'optimisation de la largeur des données au moment de l'exécution. En outre, cette technique doit modifier le format des paquets de transmission et ne peut être utilisée dans les systèmes standardisés.

2.3 Principe de l'adaptation dynamique de la précision

L'objectif de notre approche d'adaptation dynamique de la précision (ADP, ou encore DPS pour *Dynamic Precision Scaling*) est d'adapter, au cours du temps, la spécification virgule fixe, et donc la taille des opérateurs et de la mémoire, en fonction des paramètres de l'environnement extérieur, afin de réduire la consommation d'énergie. Lorsque les paramètres de l'environnement extérieur évoluent, le système bascule vers une nouvelle spécification virgule fixe adaptée à la nouvelle valeur de ces paramètres externes ce qui réduit ou augmente le nombre effectif de bits pour les opérateurs arithmétiques et la mémoire. Dans notre approche, différentes spécifications en virgule fixe sont disponibles et sont déterminées lors de la phase de conception du système.

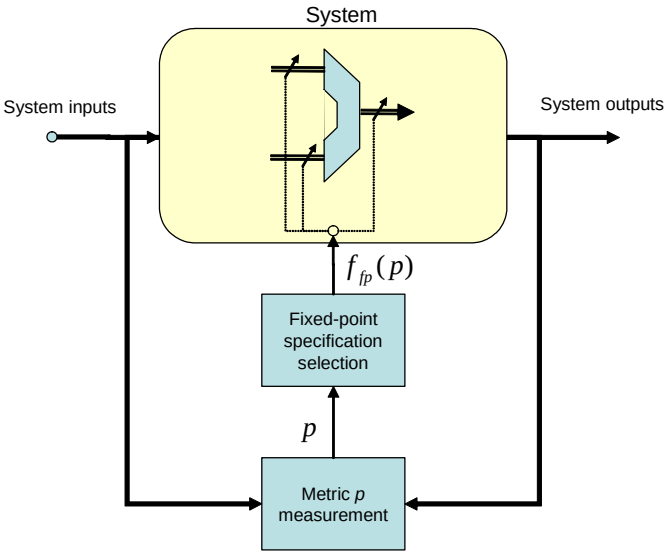


FIGURE 2.1: Principe de l'approche d'adaptation dynamique de la précision

Le principe d'un système exploitant l'adaptation dynamique de la précision est présenté à la Figure 2.1. L'architecture utilisée pour implanter le système doit posséder la capacité de réaliser les calculs sur des largeurs de données différentes et la réduction de la largeur des données doit permettre de diminuer la consommation d'énergie ou d'augmenter les performances ce qui permettrait de réduire la tension d'alimentation du circuit. Les supports d'exécution candidats pour implanter ce type de technique sont présentés dans la section 2.4.

Pour adapter la spécification virgule fixe aux paramètres de l'environnement extérieur, une métrique p décrivant les conditions extérieures est utilisée. Ce paramètre est déterminé à l'intérieur du système numérique, par la mesure du signal d'entrée et/ou du signal de sortie. La spécification virgule fixe est sélectionnée en fonction de cette métrique. Soit S_{fp}, l'ensemble de toutes les spécifications virgule fixe pouvant être utilisées. Soit f_{fp}, la

fonction de définition de la spécification virgule fixe à utiliser en fonction de la métrique p est définie par

$$\begin{array}{rcl} f_{\text{fp}} : \mathbb{R} & \longrightarrow & S_{\text{fp}} \\ p & \longmapsto & f_{\text{fp}}(p) \end{array} \tag{2.1}$$

2.3.1 Métriques p

L'objectif de la métrique p est de quantifier les conditions externes au système influençant la contrainte de précision des calculs. Cette contrainte est fixée de sorte que la dégradation des performances de l'application soit limitée. Les applications retenues pour illustrer notre approche étant issues du domaine des communications numériques, la métrique p utilisée pour quantifier les conditions externes pour ce type d'applications est détaillée.

Dans un système de transmission numérique, le rapport signal sur bruit à l'entrée du récepteur influence les performances de l'application mesurées à travers le taux d'erreurs binaires. Cette valeur n'est pas mesurable et doit être estimée. L'estimation de cette valeur est primordiale pour le bon fonctionnement du système. Le choix de la technique adéquate pour réaliser cette estimation ne fait pas partie du cadre de cette thèse. Dans [120] différentes techniques d'estimation du RSB pour un canal BBGA sont comparées. Pour le récepteur WCDMA, utilisé comme application dans le chapitre 3, la technique d'estimation par apprentissage peut être utilisée. En effet, une séquence d'apprentissage (bits pilotes) est présente dans les trames du canal dédié de contrôle (DPCCH) de la norme WCDMA. Pour le récepteur QPSK, présenté dans le chapitre 3, l'estimation du RSB se base sur l'écart entre la valeur du symbole estimé avant et après décision. Le choix de l'estimateur est un compromis entre la qualité de l'estimation et la complexité de cet estimateur. En effet, la consommation d'énergie supplémentaire due à la partie d'adaptation doit être réduite au minimum pour ne pas détruire le gain d'énergie apporté par l'adaptation de la spécification virgule fixe.

Remarque : dans cette thèse, le terme E_b/N_0 est utilisé dans les différentes simulations. E_b/N_0 est égal à la valeur du RSB divisé par l'efficacité spectrale. Par exemple, dans le cas de la modulation QPSK, E_b/N_0 est égal à $\text{RSB}/\log_2 4$. Dans le cas d'une transmission à étalement de spectre avec un facteur d'étalement SF, E_b/N_0 est égal à $\text{RSB} \times \text{SF}$.

2.3.2 Contraintes de performance

Pour une implémentation en virgule fixe, une précision minimale des calculs doit être respectée pour garantir que les performances du système sont maintenues. Dans un système de transmission numérique, le critère de performance est le taux d'erreurs binaires. Les performances sont évaluées afin que l'utilisation de l'arithmétique en précision finie ne modifie pas les performances du système en précision infinie (TEB_0) de plus de ϵ. Soit P_b, la puissance du bruit de quantification, le critère de performance peut être donc formulé de la manière suivante :

$$\text{TEB}_0 \leq \text{TEB}(P_b) \leq (1+\epsilon)\text{TEB}_0 \tag{2.2}$$

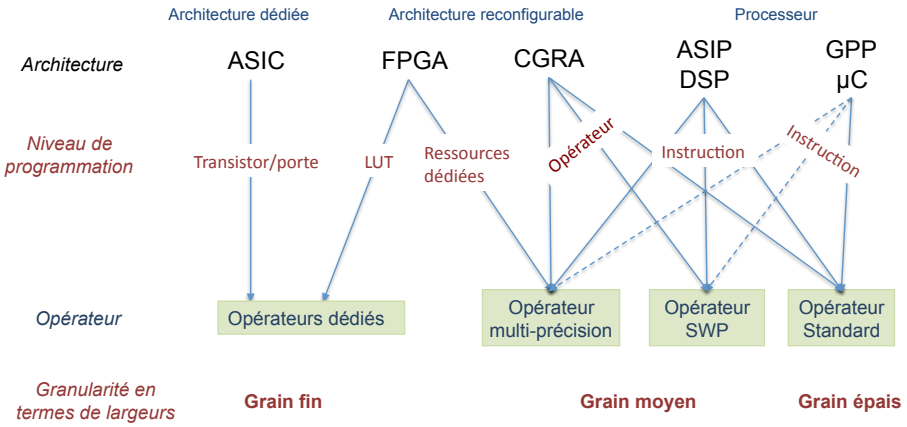


FIGURE 2.2: Granularité en termes de largeur des données

2.4 Support d'exécution

2.4.1 Granularité en termes de largeur des données

Différentes plateformes peuvent être considérées pour l'implantation d'applications de traitement du signal au sein de systèmes embarqués. Ces plateformes possèdent différentes caractéristiques en termes de coût, de consommation d'énergie et de flexibilité. Les processeurs à usage général (GPP pour *General Purpose Processor*) permettent d'implanter tous les types d'algorithmes, mais leur architecture n'est pas spécialisée pour un domaine particulier tel que celui du traitement numérique du signal (TNS). Les performances en termes d'efficacité énergétique sont relativement faibles pour les algorithmes du domaine du TNS.

Les circuits dédiés (ASIC pour *Application Specific Integrated Circuit*) sont conçus et optimisés pour une application donnée. En conséquence, les performances en termes de temps de calcul, d'efficacité énergétique et de ratio coût performance sont très bonnes. Mais ces circuits offrent une flexibilité très réduite, voire nulle. Les processeurs DSP et les cœurs de DSP sont conçus pour implanter efficacement les algorithmes spécifiques au domaine du traitement du signal. Ces architectures fournissent un bon compromis entre la flexibilité et l'efficacité énergétique. Pour obtenir un niveau de flexibilité relativement important et des performances élevées, différents types d'architectures reconfigurables ont été proposés. Les composants programmables de type FPGA (*Field–Programmable Gate Array*) fournissent une reconfiguration au niveau logique. Cette reconfiguration est réalisée au niveau des éléments logiques de l'architecture et de leurs interconnexions. La reconfiguration au niveau fonctionnel (CGRA pour *Coarse Grained Reconfigurable Architectures*) permet de modifier le flot des données au sein de l'architecture. Les interconnexions entre les ressources et la fonctionnalité de ces ressources peuvent être reconfigurées.

Ces différentes plateformes fournissent différentes caractéristiques en termes de granularité des largeurs de données supportées. Trois catégories de granularité de largeurs peuvent être distinguées comme présentées à la figure 2.2. Cette classification dépend du niveau auquel l'architecture est programmée ou configurée.

Pour les architectures à granularité de largeur importante, uniquement une largeur de

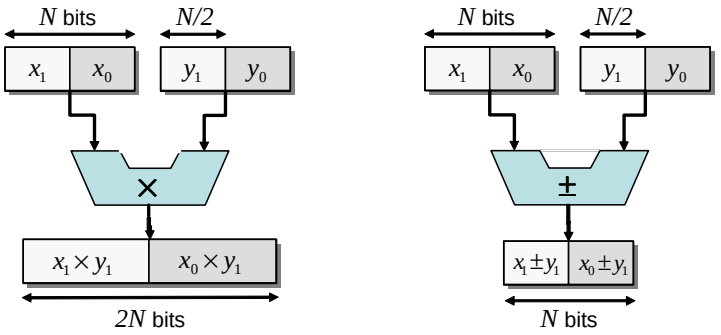


FIGURE 2.3: opérateur supportant le parallélisme de données

donnée est supportée pour chaque type d'opérateur. Dans ce cas, l'architecture ne fournit aucun mécanisme supportant une autre largeur de donnée même pour des opérations en multi-précisions.

Pour les architectures à granularité de largeur fine, toutes les largeurs ou une majorité des largeurs comprises dans un intervalle donné, sont supportées. Classiquement le pas entre deux largeurs consécutives supportées varie de un à trois bits. Cette granularité est obtenue lorsque l'architecture peut être configurée ou conçue au niveau bit comme pour les ASICs ou pour les FPGA lorsque les éléments logiques sont utilisés.

Pour les architectures à granularité de largeur moyenne, pour chaque opérateur, plusieurs largeurs sont supportées. Cette granularité est supportée dans de nombreux processeurs ou des architectures reconfigurables au niveau opérateur (CGRA Coarse Grained Reconfigurable Architectures) lorsque que celles-ci ciblent des applications de traitement du signal et dans les FPGA à travers les ressources dédiées. Le support de largeurs multiples est obtenu en utilisant les techniques de parallélisme au niveau données (SWP : *Sub-Word Parallelism*) ou de multi-precisions (MP). Ce niveau de granularité est obtenu pour les architectures programmables au niveau opérateur pour les CGRA et au niveau instruction pour les processeurs.

Dans la suite de cette section les différentes techniques permettant de fournir une granularité moyenne ou fine sont présentées.

Parallélisme de données

L'idée principale de l'exploitation du parallélisme de données est d'utiliser un opérateur pour exécuter en parallèle sur celui-ci des opérations sur des données de taille plus faible. Cette technique SWP (*Sub-Word Parallelism*) [1] divise un opérateur manipulant des données de largeur W_{SP} bits afin de pouvoir exécuter en parallèle k opérations sur des fractions de mot de largeur W_{SP}/k [46] [3]. Ce concept est illustré à la figure 2.3 pour les opérations de multiplication et d'addition/soustraction. Pour un additionneur de largeur W_{SP}, la réalisation de k additions en parallèle nécessite de ne pas propager la retenue entre deux fractions de mot. Le coût d'une opération est diminué d'un facteur k, mais en contrepartie la précision des données est nettement plus faible. Cette technique peut être utilisée pour les opérations de multiplication, d'addition, de soustraction et de décalage.

1. Le terme SIMD est aussi employé pour dénommer cette technique.

Le traitement des données en parallèle au sein d'un même opérateur engendre des contraintes au niveau du placement des données en mémoire. Ces données doivent être alignées correctement les unes par rapport aux autres. Les données doivent être rangées en mémoire en fonction de l'ordre de traitement de celles-ci.

Les architectures supportant cette technique doivent fournir des capacités de concaténation et d'extraction de fractions de mot. Les premières permettent de concaténer plusieurs fractions de mot de largeur W_{SP}/k au sein d'un mot de largeur W_{SP} et les secondes réalisent l'extraction d'une fraction de mot située au sein d'un mot de largeur plus importante. L'efficacité d'une solution utilisant les opérations SWP réside dans la limitation du surcoût lié aux instructions de concaténation ou d'extraction. En effet, ce surcoût ne doit pas annihiler le gain obtenu par la réalisation d'opérations en parallèle.

Pour illustrer les capacités SWP présentes au sein d'architectures récentes, le jeu d'instructions dédié aux opérations arithmétiques du TMS320C64x+ a été analysé. Les différentes instructions disponibles pour l'addition et la multiplication sont résumés dans le tableau 2.1. Les termes W_{in1}, W_{in2} représentent respectivement la largeur des opérandes d'entrée et W_{out} correspond à la largeur du résultat de l'opération. Soient T_{lat} et T_{th} respectivement la latence et la cadence de l'instruction. Cette cadence est de 1 cycle pour chaque instruction, ainsi, une nouvelle instruction peut être démarrée à chaque cycle.

Pour l'addition, trois instructions manipulant différentes largeurs sont disponibles. une, deux et quatre additions peuvent être réalisées sur des données de largeur respectivement 32, 16 ou 8-bits. Pour la multiplication, sept instructions sont disponibles. Pour certaines instructions (`MPY2IR`, `MPYxIR`), le résultat de l'opération est tronqué et ainsi, la largeur de celui-ci est plus importante que la largeur avec une précision complète. Le suffixe `R` indique que le mode de quantification associé à la sortie est l'arrondi. Deux instructions (`MPYxI`, `MPYxIR`) manipulent les mêmes largeurs de données mais les résultats sont stockés sur 48 ou 32 bits. Dans le premier cas, deux registres sont nécessaires pour stocker le résultat de l'opération et dans le second uniquement un registre est utilisé mais les 16 bits de poids faible sont éliminés. Les opérations utilisant ce dernier résultat manipuleront des données stockées sur un seul registre et ainsi, nécessiteront moins de temps.

Pour manipuler les données, cette architecture propose deux types d'instructions. Pour concaténer les sous-mots au sein d'un mot de largeur W_{SP}, six instructions `PACK` sont disponibles. Pour extraire un sous-mot d'un registre deux instructions `UNPACK` sont fournies. Ces instructions distribuent les sous-mots dans plusieurs registres. Ces instructions sont nécessaires si une quantification du résultat d'une opération est réalisée afin de pouvoir traiter par la suite des données de taille plus faible. Ces instructions augmentent le temps d'exécution du code et diminuent l'accélération apportée par la technique SWP.

Multi-précision

L'idée principale des opérations multi-précisons (MP) est d'utiliser un ensemble d'opérateurs manipulant des données en simple précision sur W_{SP} bits pour effectuer des opérations sur des données ayant une largeur supérieure à W_{SP} bits. L'addition de deux données sur W_{MP} bits nécessite N_{MP}^{add} additions en simple précision (SP) sur W_{SP}^{add} bits. Le terme N_{MP}^{add} est défini comme

$$N_{MP}^{add} = \left\lceil \frac{W_{MP}}{W_{SP}} \right\rceil . \tag{2.3}$$

Instructions	W_{in1}	W_{in2}	W_{out}	k	T_{lat}	T_{th}
ADD4	8	8	8	4	1	1
ADD2	16	16	16	2	1	1
ADD	32	32	32	1	1	1
MPY4	8	8	16	4	4	1
MPY2	16	16	32	2	4	1
MPY2IR	16	32	32	2	4	1
MPYx	16	16	32	1	1	1
MPYxI	16	32	48	1	4	1
MPYxIR	16	32	32	1	4	1
MPY32	32	32	32	1	4	1

TABLE 2.1: Jeu d'instructions du TMS320C64x+ utilisant la technique SWP

Cette addition MP nécessite une addition SP A_0 pour les W_{SP}^{add} bits de poids faible et $N_{MP}-1$ additions SP avec retenue entrante $(A_1 \dots A_{N_{MP}})$ pour les autres bits. La retenue entrante de l'addition A_i est connectée à la retenue de sortie de l'addition A_{i-1}.

La multiplication MP de W_{MP1} bits par W_{MP2} bits nécessite N_{MP}^{mult} multiplications SP de W_{SP}^{mult} bits. Le terme N_{MP}^{mult} est défini comme

$$N_{MP}^{mult} = \left\lceil \frac{W_{MP1}}{W_{SP}^{mult}} \right\rceil . \left\lceil \frac{W_{MP2}}{W_{SP}^{mult}} \right\rceil . \tag{2.4}$$

Les multiplieurs sur N_{MP}^{mult} bits doivent réaliser des opérations sur des données signées et non-signées en fonction de la position du mot SP au sein de la donnée MP. $N_{MP}^{mult} - 1$ additions sur des données de $2.W_{SP}^{mult}$ bits sont nécessaires pour additionner les résultats des multiplications SP. De plus, pour aligner les résultats des multiplications SP des opérations de décalage sont nécessaires.

Comme indiqué dans les équations 2.3 and 2.4, les calculs en MP augmentent le nombre d'opérations à effectuer et en particulier la multiplication qui conduit à nombre élevé d'opérations à réaliser. Cette technique permet de traiter efficacement des opérations asymétriques. Ainsi, si la largeur des opérandes d'une opération est optimisée, le coût de l'implantation peut être réduit par rapport à une approche symétrique.

Par exemple, dans le cas d'une architecture SP possédant un multiplieur 16×16 bits, le nombre de multiplications SP nécessaires pour une multiplication de 32×16 bits est égal à la moitié du nombre de multiplications SP nécessaires pour une multiplication symétrique de 32×32 bits.

Deux types d'implantation des techniques multi-précisions peuvent être distinguées. Une implantation spatiale d'une opération MP peut être considérée. Dans ce cas, plusieurs opérateurs sont utilisés en parallèle. Ces opérateurs sont inter-connectés entre eux afin de composer un macro-opérateur permettant de traiter des données ayant une largeur plus importante. Un pipeline peut être mis en œuvre pour augmenter la cadence des

traitements. Cette technique est très utilisée dans le cas d'architectures reconfigurables et plus particulièrement pour les FPGA à travers l'utilisation des ressources dédiées.

Une implantation temporelle d'une opération MP peut être considérée lorsque le nombre de ressources disponibles au sein de l'architectures est limité. Ainsi, un partage de l'opérateur au cours du temps est effectué pour pouvoir exécuter la suite d'opérations. L'opération MP est décomposée en une suite d'opérations SP exécutées séquentiellement. Un microcode est associé à chaque opération MP. Plusieurs cycles sont nécessaires pour calculer une opération MP. Cette technique est utilisée dans le cas des processeurs lorsque le nombre d'opérateurs disponibles est limité.

Pour la première génération de processeurs DSP possédant des capacités de décalage limitées, le nombre de cycles pour calculer une multiplication en double précision est assez important. Par exemple, pour le DSP 56000, 26 cycles sont nécessaires. Pour la famille de DSP C5000 de Texas Instruments, une multiplication en DP est proposée au sein de la bibliothèque *dsplib*. Cette multiplication nécessite 7 cycles au lieu d'un seul cycle dans le cas d'une multiplication en SP. Cette multiplication en DP utilise une instruction de multiplication-accumulation (MAC) pour des données signées ou non-signées. Dans certains DSP plus récents tel que le BlackFin d'Analog Device, les capacités de parallélisme de l'architecture permettent d'obtenir une multiplication en DP en seulement 3 cycles [94]. De plus, ce temps d'exécution peut être réduit à 2 cycles en réalisant une multiplication en DP tronquée. Dans ce cas, les bits de poids faible du résultat ne sont pas calculés. La multiplication en SP entre les deux mots de poids faible des opérandes d'entrée n'est pas réalisée.

Différents opérateurs reconfigurables utilisant des opérations MP ont été proposés. Ils fournissent une flexibilité intéressante en termes de largeur des données supportées. Dans [92], un réseau de multiplieurs 4×4 bits est utilisé pour proposer une multiplication 64×64 bits, quatre multiplications 32×32 bis, 16 multiplications 16×16 bits and 64 multiplications 8×8 bits. Dans [146], le même type de réseau de multiplieurs 4×4 bits permet d'obtenir 2^j multiplications symétriques sur des données de 2^{7-j} bits avec $j \in [0; 4]$. La latence de la multiplication est égale à $5 - j$ cycles d'horloge. Dans [67], les opérations en MP sont réalisées à partir d'un multiplieur 8×8 bits. Des multiplieurs efficaces en termes de latence doivent être conçus afin d'obtenir une latence raisonnable pour l'opération en MP.

Dans le FPGA Startix d'Altera, chaque *DSP Block* est composé de quatre multiplieurs câblés 18×18 bits. Les techniques SWP et MP sont utilisées en fonction de la largeur des données. En utilisant la technique MP, une multiplication 36×36 bits peut être obtenue. En utilisant la technique SWP, le multiplieur câblé est décomposé en deux pour pouvoir réaliser deux multiplications 9×9 bits. Ainsi, chaque *DSP Block* peut être configuré pour réaliser huit multiplications 9×9 bits. Les architectures FPGA de Lattice basées sur les *sysDSP Blocks* fournit les mêmes caractéristiques que l'architecture Stratix d'Altera. L'architecture *Variable-Precision DSP Architecture* [132], supporte en natif différentes largeurs de données. Cette architecture fournit des opérations de faible précision avec trois multiplications indépendantes de 9×9 bits, des opérations de précision moyenne avec deux multiplications de 18×18 bits et des opérations de précision plus élevée avec une multiplication de 27×27 ou de 18×36 bits.

Opérateurs customisés

L'implantation d'applications au sein d'un ASIC ou d'un FPGA nécessite de concevoir et développer des opérateurs spécifiques de largeurs arbitraires. Étant donné que les ASIC sont définis au niveau porte logique et les FPGA au niveau élément logique, ces opérateurs permettent d'obtenir une granularité en termes de largeur d'un bit. La réduction d'un bit pour un opérateur doit permettre de réduire le coût de celui-ci. En conséquence, ce type d'architecture permet d'obtenir de nombreux compromis entre le coût de l'opérateur et la précision (largeur des données) fournie par celui-ci. Ainsi, la frontière de Pareto du coût de l'implantation en fonction de la précision des calculs conduit à une diversité de points.

Pour les applications de TNS, les opérations les plus importantes sont l'addition, la soustraction et la multiplication. Différentes structures ont été proposées pour les additionneurs [157] et pour les multiplieurs. Dans le cas d'une cible de type ASIC, la diversité de ces structures permet d'obtenir différents compromis en termes de coût de l'implantation (surface, consommation d'énergie) et de latence de l'opérateur [69, 16]. Pour les FPGA, les architectures proposées favorisent les additionneurs basés sur une structure à propagation de retenue [31]. En effet, de nombreux FPGA intègrent une chaine rapide de propagation de retenue.

Opérateurs à largeur variable

Dans cette section, nous avons regroupé les techniques proposées pour obtenir un opérateur dont le coût en termes de consommation d'énergie dépend de la largeur des données manipulées. Ces opérateurs vont exécuter une seule opération par cycle en utilisant un seul opérateur.

Dans [11], un multiplieur capable de réaliser des opérations sur différentes largeurs est proposé. Ce circuit est conçu grâce à un détecteur de zéro, un multiplexeur et quatre multiplieurs séparés. En fonction de la largeur des opérandes, déterminée à partir du nombre de zéro détectés, le multiplieur le plus adéquat est sélectionné. Pour cette approche, l'adaptabilité de l'opérateur et ainsi la réduction potentielle d'énergie est obtenue au détriment de l'augmentation de la surface. En effet, pour pouvoir traiter N largeurs différentes, N multiplieurs sont utilisés. La combinaison des largeurs des différents multiplieurs est déterminée en fonction des caractéristiques de l'application ciblée. Dans l'exemple traité, une combinaison de quatre multiplieurs sur 9, 11, 14, et 16 bits est utilisée et cette combinaison permet de réduire de 60% de la consommation d'énergie par rapport à un multiplieur 16 bits traditionnel.

Dans, [119], un compromis entre la qualité d'image et la consommation d'énergie est proposé pour une application correspondant à une DCT. Cette DCT est réalisée à l'aide d'un ensemble d'additions, de soustractions et de décalages. La structure utilisée pour les additionneurs est de type *carry save*. Afin de réduire la consommation d'énergie, les bits LSB non utilisés sont mis à zéro et les additionneurs 1 bit correspondant sont donc éteints.

Dans [62], l'auteur analyse le gain en consommation d'énergie lié à la réduction de la largeur des données. Cette réduction de la largeur permet de réduire le taux d'activité au sein d'un multiplieur. Un multiplieur 32×32 bits est utilisé et l'activité au sein du multiplieur est analysée pour deux techniques. La première technique consiste à aligner les opérandes sur le bit le plus significatif (MSB) et de compléter les bits les moins significatifs

non utilisés par des zéros. La seconde technique consiste à aligner les opérandes sur le bit le moins significatif (LSB) et de compléter les bits les plus significatifs par le bit de signe. Les cas du recodage de Booth et d'arbres de réduction de produits partiels de Wallace sont testés. Ces techniques permettent de réduire le taux d'activité et ainsi la puissance dynamique par rapport à une multiplication 32×32 bits.

2.4.2 Définition du support d'exécution

L'architecture permettant d'implanter l'approche d'adaptation dynamique de la précision (ADP) en vue d'optimiser la consommation d'énergie doit posséder des caractéristiques particulières présentées ci-dessous. L'architecture doit permettre de supporter différentes configurations de spécification virgule fixe. Ces différentes spécifications virgule fixe doivent conduire à une diversité de compromis entre la précision des calculs et la consommation d'énergie. Cette architecture doit être capable de basculer d'une configuration à une autre avec une pénalité en termes de temps et de consommation d'énergie faible.

Faible consommation d'énergie L'objectif de notre travail étant d'adapter dynamiquement la précision en vue d'optimiser la consommation d'énergie, il est indispensable que l'architecture considérée soit efficace d'un point vue énergétique. Il y a peu d'intérêt à réaliser de l'ADP sur une architecture dont la consommation de base est élevée. Il est préférable d'utiliser une architecture ayant une consommation de base plus faible sans ADP afin d'obtenir de meilleurs performances en termes de consommation d'énergie.

Reconfigurabilité des traitements Une configuration particulière est associée à chaque spécification virgule fixe. L'architecture doit permettre de changer de configuration rapidement. Ce changement de configuration ne doit pas entraîner une surconsommation d'énergie annihilant le gain de consommation d'énergie lié au basculement de configuration.

Pour les processeurs chaque configuration peut correspondre à une tâche ou à une fonction. Ainsi, en fonction de la métrique p définie dans la section 2.3.1, le système change de fonction ou de tâche. Pour les architectures reconfigurables, chaque spécification virgule fixe correspond à une configuration matérielle donnée. Pour les FPGA, le temps et la consommation d'énergie liés au chargement d'un nouveau *bitstream* n'est pas négligeable, ainsi le basculement entre les différentes configurations ne doit pas être réalisé trop souvent. Pour les CGRA, le passage d'une configuration à une autre correspond à la modification des commandes des opérateurs et du réseau d'interconnexion. Ainsi, le temps de changement de configuration et la consommation d'énergie engendrée par celui-ci peut être raisonnable. Par défaut, les ASIC étant des architectures spécialisées à une application particulière, ils ne fournissent pas de flexibilité. Cependant, une architecture spécialisée à l'application considérée, mais possédant des opérateurs configurables en termes de largeur, peut être envisagée. Ces opérateurs traitent une seule donnée mais pourraient supporter plusieurs largeurs. Pour les largeurs plus faibles, la partie de l'opérateur non utilisée est mise en veille afin de réduire la consommation. Ce concept doit être étendu aux bus de données et à la mémoire pour aboutir à une architecture offrant une diversité intéressante de consommation d'énergie en fonction des largeurs traitées.

Flexibilité en termes de largeurs des données supportées L'architecture doit fournir un minimum de diversité en termes de types de données supportés afin de pouvoir implanter plusieurs spécifications virgule fixe conduisant à des précisions de calcul différentes. D'après l'analyse présentée dans la section 2.4.1, l'architecture doit posséder une granularité de largeurs fine ou moyenne.

Support pour l'adaptation dynamique de la précision La première solution pour réaliser de l'ADP correspond à un processeur ou une architecture reconfigurable au niveau opérateur. Cette architecture doit posséder des opérateurs ayant des capacités de multi-précisions ou de parallélisme au niveau donnée et ayant été conçus afin de minimiser la consommation d'énergie. Cette solution permet d'avoir une granularité moyenne en termes de largeurs supportées, la possibilité de passer rapidement d'une configuration virgule fixe à une autre en changeant la configuration ou les fonctions utilisées. Pour illustrer ce type d'architectures, nous pouvons utiliser l'architecture développée dans le cadre du projet ROMA [100]. L'opérateur reconfigurable utilisé au sein de cette architecture possède des capacités de parallélisme au niveau des données.

La seconde solution correspond à un FPGA basse consommation tel que le FPGA Igloo proposé par la société Actel. Cette solution permet d'obtenir une granularité plus fine mais en contrepartie, les changements de configurations ne doivent pas être trop fréquents afin de ne pas être pénalisés par les temps et l'énergie consommée par la reconfiguration.

La troisième solution est de concevoir une architecture dont la largeur du chemin de données et de la mémoire est configurable. La mise en œuvre de techniques de *power gating* permettrait de couper l'alimentation des parties du circuit non utilisées [147].

2.5 Conclusion

La dynamique des données au sein d'une application et la contrainte de précision, permettant de limiter la dégradation des performances, peut évoluer au cours du temps en fonction de la nature du signal en entrée du système. Ces variations sont liées à la modification des conditions de l'environnement extérieur au système. La prise en compte de ces variations peut être une opportunité pour réduire la consommation d'énergie du système. Après un état de l'art des approches existantes, le concept d'adaptation dynamique de la précision a été présenté puis le support d'exécution permettant d'implanter ce type d'approche a été défini.

Chapitre 3

Adaptation dynamique de la précision : application aux systèmes de communication sans-fil

Dans ce chapitre, le principe d'adaptation dynamique de la précision est appliqué aux systèmes de communication sans fil. Pour cela, l'expression analytique de la dynamique et du critère de précision est déterminée en fonction du rapport signal à bruit à l'entrée du système.

Dans un premier temps, un récepteur QPSK est étudié. La relation entre le taux d'erreur binaire (TEB ou encore BER pour *Binary Error Rate*) et le rapport signal à bruit de quantification est déterminée en fonction du rapport signal à bruit en entrée du récepteur. Deux cas sont considérés : une quantification unique et des quantifications multiples. L'erreur de quantification est considérée comme un bruit uniformément distribué dans le premier cas et comme un bruit gaussien dans le second cas. Ensuite, ce résultat est appliqué aux systèmes d'étalement de spectre par séquence directe (DS-CDMA). Dans un troisième temps, une application concrète des systèmes DS-CDMA est étudiée au travers d'un récepteur WCDMA pour les télécommunications de troisième génération.

3.1 Chaîne de transmission QPSK

3.1.1 Introduction

Pour illustrer simplement l'intérêt de l'adaptation dynamique de la précision, un premier exemple de système de communication numérique est considéré. Ce système, illustré dans la figure 3.1, se compose d'un émetteur et d'un récepteur utilisant la modulation QPSK. Le signal est multiplexé en deux voies I, Q par un convertisseur série–parallèle et démultiplexé après le décodage. Le canal de transmission est modélisé par un bruit blanc gaussien additif (BBGA, ou encore AWGN pour *Additive White Gaussian Noise*) [1] avec

1. BBGA est un modèle de canal de transmission où un bruit blanc est linéairement ajouté au signal. L'amplitude du bruit suit une distribution gaussienne.

E_b/N_0 variant entre 0 dB et 10 dB. Aucun codage de canal n'est utilisé. Le signal d'entrée du récepteur et le taux d'erreur binaire sont observés.

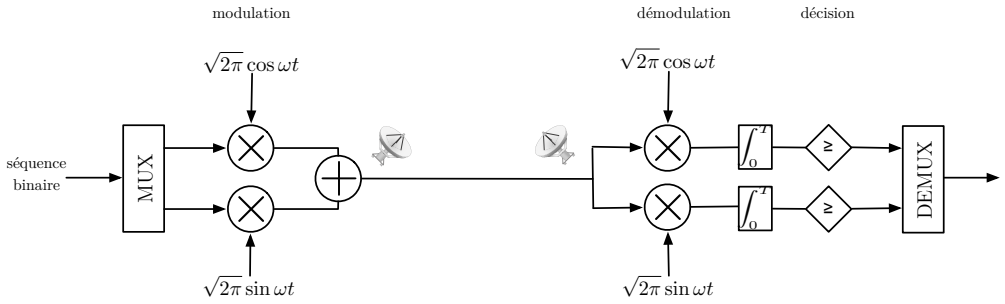


FIGURE 3.1: Graphe flot d'une chaîne de transmission QPSK.

L'objectif de l'analyse suivante est de déterminer le nombre minimal de bits nécessaires pour la partie entière et la partie fractionnaire. La spécification virgule fixe doit garantir l'absence de débordement et maintenir les performances en termes de taux d'erreur binaire.

3.1.2 Estimation de la dynamique

Le but de cette partie est de déterminer la valeur minimale du nombre de bits pour la partie entière permettant de garantir l'absence de débordement. Dans le cas d'un canal BBGA, le signal d'entrée $y(k)$ du récepteur correspond à la somme des symboles émis s et du bruit c associé au canal tel que

$$y(k) = s(k) + c(k), \tag{3.1}$$

où $s(k)$ est l'ensemble des symboles modulés avec

$$s(k) \in \{\pm 1 \pm i\}. \tag{3.2}$$

La dynamique d'une donnée correspond à l'intervalle regroupant l'ensemble des valeurs prises par cette donnée au cours du temps. Les équations (3.1) et (3.2) montrent que plus le niveau de bruit $c(k)$ est grand, plus grande sera la dynamique du signal reçu $s(k)$. Par exemple, la dynamique à 0 dB est deux fois plus importante que celle obtenue pour de très faibles niveaux de bruit. La dynamique obtenue pour différents niveaux de RSB est présentée dans la figure 3.2. La dynamique du signal d'entrée diminue lorsque le rapport signal sur bruit augmente. Il diminue de 3,2 (à 0 dB) à 1,7 (à 10 dB). Entre ces deux valeurs de RSB, un bit à l'entrée du récepteur peut être économisé.

3.1.3 Analyse de la précision

Le but de cette partie est de déterminer la valeur minimale du nombre de bits pour la partie fractionnaire garantissant un taux d'erreur binaire proche de celui obtenu avec un calcul en précision infinie. Dans ce chapitre, uniquement le cas de la quantification par arrondi est traité.

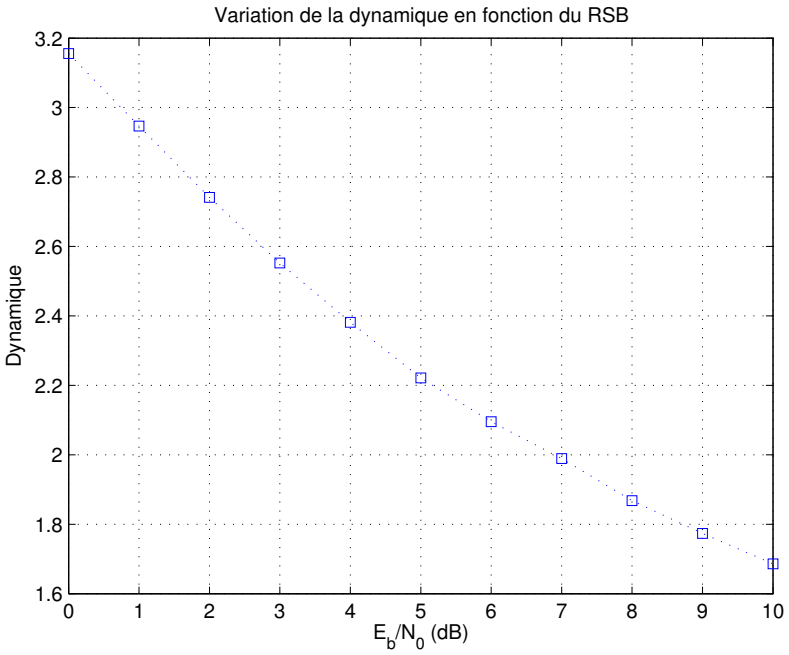


FIGURE 3.2: Analyse de la dynamique pour la détection de symboles dans le cas d'un canal BBGA et la modulation QPSK.

Le canal $c(k)$ est supposé blanc gaussien avec une variance de bruit $\sigma_c^2 = \frac{N_0}{2E_b}$. L'expression de la densité de probabilité $f_c(x)$ de $c(k)$ est

$$f_c(x) = \frac{1}{\sigma_c\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_c^2}}. \tag{3.3}$$

Sans erreur de quantification, la probabilité d'erreur binaire de cette transmission est déterminée par

$$\text{TEB} = Q(\sigma_c^{-1}) = \frac{1}{2}\,\text{erfc}(\frac{1}{\sigma_c\sqrt{2}}), \tag{3.4}$$

avec la fonction Q [122] définie par

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt = \frac{1}{2}\,\text{erfc}(\frac{x}{\sqrt{2}}). \tag{3.5}$$

Cas d'une quantification unique

Le signal analogique $y(k)$ est quantifié avec le pas de quantification $q = 2^{-m}$, m le nombre de bits de la partie fractionnaire, par le convertisseur analogique – numérique du récepteur. L'expression de la densité de probabilité $f_q(x)$ du bruit de quantification est alors

$$f_q(x) = \frac{1}{q}\mathbf{I}_{[\frac{-q}{2},\frac{q}{2}]}(x), \tag{3.6}$$

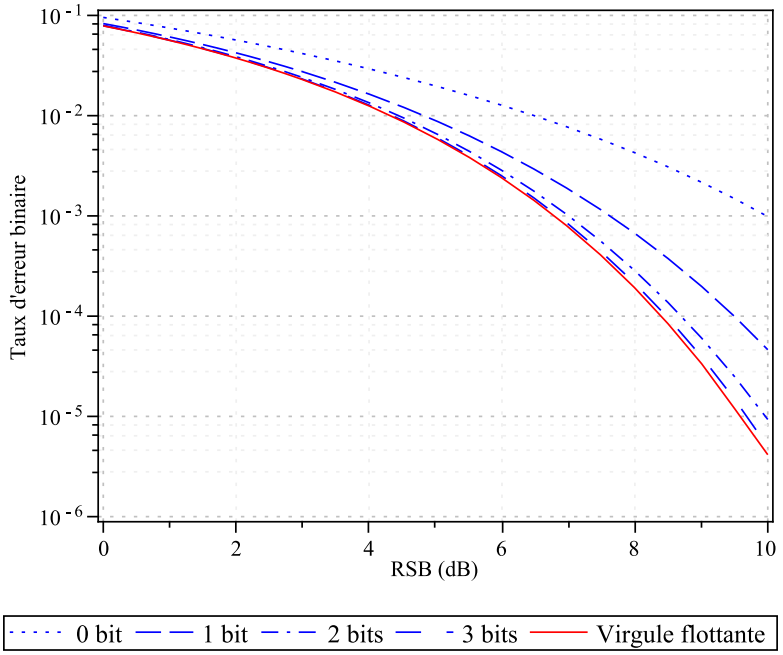


FIGURE 3.3: Valeur théorique du TEB d'une modulation QPSK avec un canal BBGA pour différentes quantifications et différents RSB.

avec $\mathbf{I}$ la fonction indicatrice définie par

$$\mathbf{I}_{\mathcal{D}}(x) = \begin{cases} 0 & \text{si } x \notin \mathcal{D} \\ 1 & \text{si } x \in \mathcal{D} \end{cases} . \tag{3.7}$$

Ainsi, la densité de probabilité du bruit global correspondant à la somme du bruit du canal et du bruit de quantification est

$$f_n(x) = f_c * f_q(x) = \int_{-\infty}^{\infty} f_c(t) f_q(x-t) dt \tag{3.8}$$

$$= \frac{1}{2q}(\text{erf}\,\frac{x+\frac{q}{2}}{\sqrt{N_0}} - \text{erf}\,\frac{x-\frac{q}{2}}{\sqrt{N_0}}). \tag{3.9}$$

La fonction de répartition du bruit vaut quant à elle

$$F_n(x) = \int_{-\infty}^{x} \frac{1}{2q}(\text{erf}\,\frac{t+\frac{q}{2}}{\sqrt{N_0}} - \text{erf}\,\frac{t-\frac{q}{2}}{\sqrt{N_0}}) dt. \tag{3.10}$$

f_n et F_n sont des fonctions dépendantes de f_c et f_q, elles prennent donc σ_c (ou N_0, ou encore RSB) et q (ou bien, m) comme paramètres. Pour des modulations de type BPSK et QPSK, la probabilité d'erreur binaire pour une quantification sur m bits pour la partie fractionnaire et pour un rapport signal à bruit RSB donné, est définie par

$$\text{TEB} = 1 - F_n(1). \tag{3.11}$$

Il n'existe pas d'expression analytique simple de l'équation (3.11), ce problème est résolu numériquement. La figure 3.3 montre les valeurs obtenues par évaluation de l'expression analytique (3.11) pour le nombre de bits de la partie fractionnaire $m \in \{0, 1, 2, 3\}$

et pour la précision infinie. Le critère de précision est défini avec un paramètre ϵ ($\epsilon \ll 1$) de telle sorte que la condition

$$\text{TEB} < (1+\epsilon)\text{TEB}_0 \tag{3.12}$$

soit respectée. TEB_0 représente le taux d'erreur binaire en précision infinie.

De cette analyse, la largeur théorique m de la partie fractionnaire d'une modulation QPSK peut être déduite des équations (3.11) et (3.12) et est présentée dans la figure 3.4 pour différentes valeurs du RSB et pour deux valeurs de ϵ. Cette largeur augmente avec le RSB et jusqu'à trois bits à l'entrée du récepteur peuvent être économisés en fonction des conditions du canal.

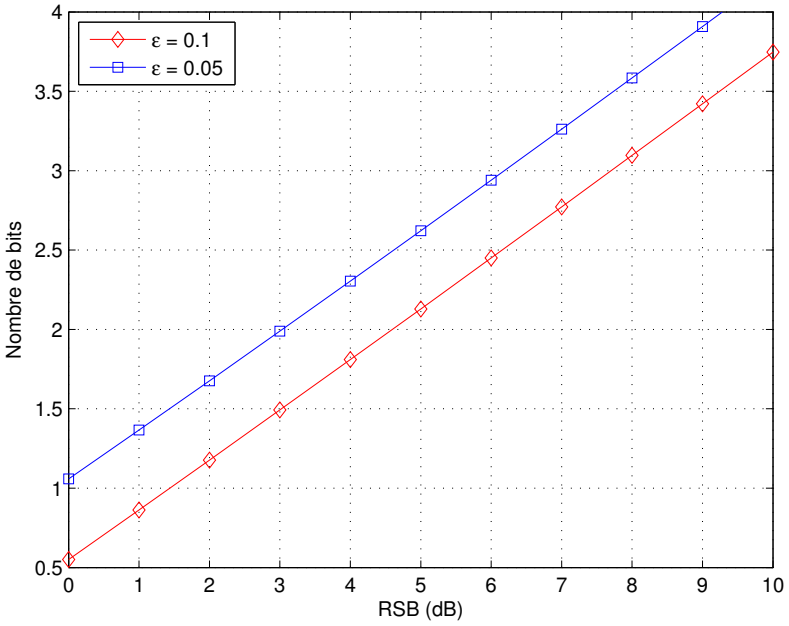


FIGURE 3.4: Largeur théorique de la partie fractionnaire d'une modulation QPSK avec un canal BBGA pour différents critères de précision et différents RSB.

Afin de vérifier cette analyse théorique, des simulations du TEB pour une modulation QPSK avec les quantifications et les conditions différentes de canal ont été réalisées. Les résultats sont présentés dans la figure 3.5. Comme le signal émis est binaire (pour chaque partie réelle et partie imaginaire), la largeur de la partie fractionnaire est choisie à zéro, un, deux et trois bits. Encore une fois, plus le RSB est important, plus la largeur nécessaire de la partie fractionnaire est grande. La figure 3.5 confirme que le processus de démodulation a besoin pour la partie fractionnaire d'un bit à 0 dB, de deux bits à 4 dB et plus de trois bits à 10 dB. Afin d'approximer la précision infinie, le bruit de quantification doit être négligeable par rapport au bruit du canal. Ainsi, un plus grand nombre de bits est nécessaire pour la partie fractionnaire lorsque le niveau de bruit est moins important.

Avec une adaptation dynamique de la spécification virgule fixe en fonction du RSB, une réduction de la largeur peut être réalisée par rapport à une approche classique. Dans le cas d'une conception classique utilisant une approche dans le pire cas, une largeur de $2+5=7$ bits est nécessaire. Avec l'approche d'adaptation dynamique proposée, $2+1=3$ bits à 0 dB et $1+5=6$ bits à 10 dB sont nécessaires. Entre ces deux valeurs de RSB, trois

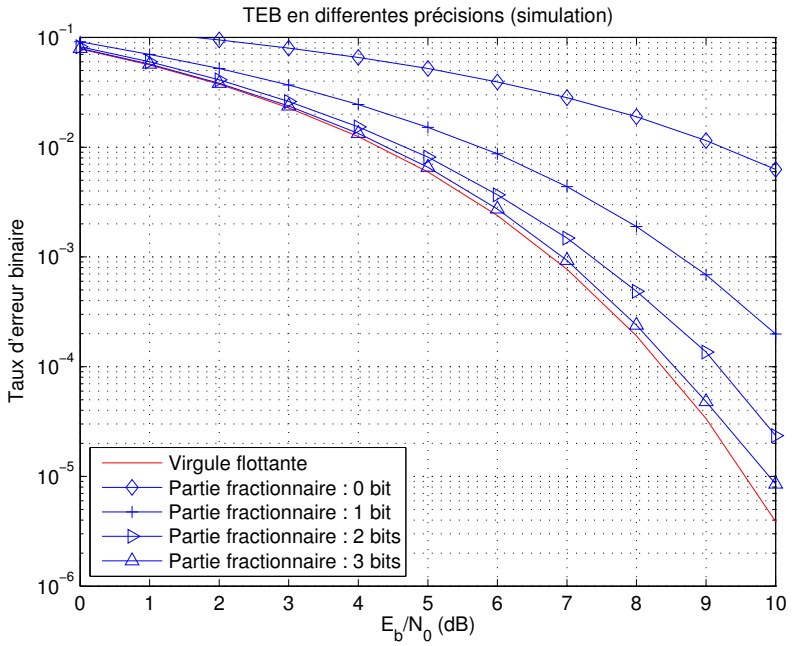


FIGURE 3.5: TEB simulé pour une modulation QPSK avec un canal BBGA pour différentes quantifications et différents RSB.

bits à l'entrée du récepteur peuvent être économisés et dans le cas d'un RSB de 0 dB, 4 bits peuvent être économisés par rapport à une approche classique.

Cas de quantifications multiples dues à des calculs en précision finie

Dans le cas général, le récepteur comprend, en plus de la quantification due au convertisseur analogique – numérique, une série de traitements et génère donc différents bruits de quantification $q_1, q_2, ..., q_K$ de puissances similaires. En raison du théorème central limite, la somme de ces bruits est alors considérée comme une gaussienne dont la densité de probabilité est

$$f_Q(x) = \frac{1}{\sigma_Q\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_Q^2}} \tag{3.13}$$

avec

$$\sigma_Q^2 = \sum_{i=1}^{K} \sigma_{q_i}^2 = \sum_{i=1}^{K} \frac{q_i^2}{12}, \tag{3.14}$$

la puissance du bruit correspondant à l'ensemble des quantifications du système.

Ainsi, le bruit global, incluant le traitement et les erreurs de quantification au niveau du récepteur possède la densité de probabilité $f_N(x)$ définie par

$$f_N(x) = \frac{1}{\sqrt{\sigma_c^2 + \sigma_Q^2}\sqrt{2\pi}} \exp \frac{-x^2}{2(\sigma_c^2 + \sigma_Q^2)} \tag{3.15}$$

ainsi qu'une fonction de répartition $F_N(x)$ donnée par

$$F_N(x) = \frac{1}{2}(1 + \text{erf}\,\frac{x}{\sqrt{\sigma_c^2 + \sigma_Q^2}\sqrt{2}}). \tag{3.16}$$

Pour une modulation de type BPSK et QPSK, l'expression de la probabilité d'erreur binaire pour un RSB donné et une puissance σ_Q^2 de l'ensemble des quantifications multiples donnée, équivalente à une quantification sur m bits, est

$$\begin{aligned} \text{TEB} = 1 - F_N(1) &= \frac{1}{2}\,\text{erfc}\,\frac{1}{\sqrt{\sigma_c^2 + \sigma_Q^2}\sqrt{2}} \\ &= Q(\frac{1}{\sqrt{\sigma_c^2 + \sigma_Q^2}}). \end{aligned} \tag{3.17}$$

L'équation (3.17) montre que dans le cas de quantifications multiples, le bruit de quantification possède les mêmes propriétés d'un bruit blanc gaussien additif. Ce bruit fait décaler la courbe TEB à gauche pour une distance de $P_b = \sigma_Q^2$ (dB).

3.1.4 Conclusion

Dans cette section, une chaîne de transmission QPSK a été étudiée. Une expression analytique a été définie pour déterminer la puissance de bruit de quantification en fonction du rapport signal sur bruit d'entrée et du taux d'erreur binaire souhaité. Le taux d'erreur binaire peut être déterminé théoriquement, pour un canal BBGA, si les bruits d'entrée et les bruits de quantification sont connus.

3.2 Systèmes à étalement de spectre DS-CDMA

Cette section présente les systèmes d'accès multiples basés sur la technique d'étalement de spectre par séquence directe DS-CDMA (pour *Direct Spread Code Division Multiple Access* en anglais) ainsi que l'adaptation dynamique des largeurs de donnée appliquée à ces systèmes.

3.2.1 Introduction

Étalement du spectre

Dans le contexte des communications numériques sans-fil, un canal unique est utilisé pour plusieurs utilisateurs qui peuvent alors accéder au canal en même temps avec le principe dénommé technique d'accès, ou « multiplexage ». Initialement, deux techniques de multiplexage étaient utilisées : le multiplexage temporel (TDMA) et le multiplexage fréquentiel (FDMA) divisent respectivement l'accès en temps et en fréquence. Ces modes d'accès allouent des ressources fixes (intervalles de temps ou bandes de fréquence) aux utilisateurs. Ces techniques d'accès multiples sont peu flexibles, possèdent une faible capacité

et subissent différents effets comme l'interférence entre symboles (ISI). Pour surmonter ces points faibles, la technique d'accès CDMA a été proposée.

La technologie d'accès multiples par répartition de codes (CDMA) est basée sur « l'étalement de spectre ». Les symboles à transmettre sont étalés sur une plus grande largeur de bande que celle strictement nécessaire. Plusieurs techniques sont employées pour étaler le signal dans un système CDMA : la séquence directe (DSSS), le saut de fréquence (FHSS), le saut de temps (THSS), ou une combinaison des précédents. Dans la suite, la technique par séquence directe utilisée pour le moment dans les systèmes DS-CDMA est considéré. Cette technique module un signal au moyen d'un code numérique à un débit supérieur à celui du signal d'information à transmettre. Ce code numérique correspond à une séquence pseudo-aléatoire, attribuée différemment aux utilisateurs. La technique WCDMA (Wideband CDMA), utilisant DSSS et présente au sein des systèmes de téléphonie mobile de troisième génération (3G), servira d'illustration à notre approche.

Les techniques d'étalement de spectre sont utilisées dans de nombreux systèmes. La norme IEEE 802.11b (Wi-Fi) utilise DSSS et FHSS. La norme IEEE 802.15.4 (ZigBee) et IEEE 802.22 (WRAN) utilise DSSS. Plusieurs systèmes de positionnement par satellites, comme le GPS des États–Unis et le Galileo de l'Europe, utilisent DSSS. Dans les systèmes de transport intelligent (ITS), la norme IEEE 802.11p pour la communication entre véhicules utilise la technique d'étalement de spectre par séquence directe.

Étalement de spectre par séquence directe (DS-CDMA)

La technique d'étalement de spectre par séquence directe est la plus simple. Elle est réalisée par la multiplication du signal d'information, de débit R_s, par un code pseudo–aléatoire (PN) ayant un débit plus élevé R_c. L'élément de base de durée T_c du code est appelé *chip*. Le nombre N_c de chips par bit d'information est égal au ratio des débits du code et du signal d'information :

$$N_c \approx \frac{R_c}{R_s} = \frac{T_s}{T_c}. \tag{3.18}$$

Ce terme N_c est appelé gain d'étalement ou gain de traitement (*processing gain*) car il représente une mesure de l'amélioration du rapport signal sur interférence. Plus précisément, le gain de traitement est le rapport entre la bande passante du signal après étalement et la bande passante d'information. Dans plusieurs applications pratiques, la bande passante du signal après étalement est égale à la largeur du lobe principal du spectre de la séquence pseudo–aléatoire (toujours sous la forme $\frac{\sin x}{x}$). Ainsi, l'approximation (3.18) peut être réalisée.

La technique DSSS est présentée à la figure 3.6. La séquence pseudo–aléatoire propre à chaque utilisateur est binaire et prend des valeurs $\{-1, 1\}$. Si la bande passante de cette séquence est beaucoup plus large que celle du signal, le signal étalé est quasiment indépendant du signal informatif. Sans connaissance de la séquence pseudo–aléatoire, il est impossible de déterminer le signal avant l'étalement. Au récepteur, supposons que le système est parfaitement synchronisé, les bits informatifs sont retrouvés après une multiplication par la même séquence pseudo–aléatoire. Le principe est illustré dans la figure 3.7. Pour simplifier, les filtres d'émission et de réception ne sont pas présentés.

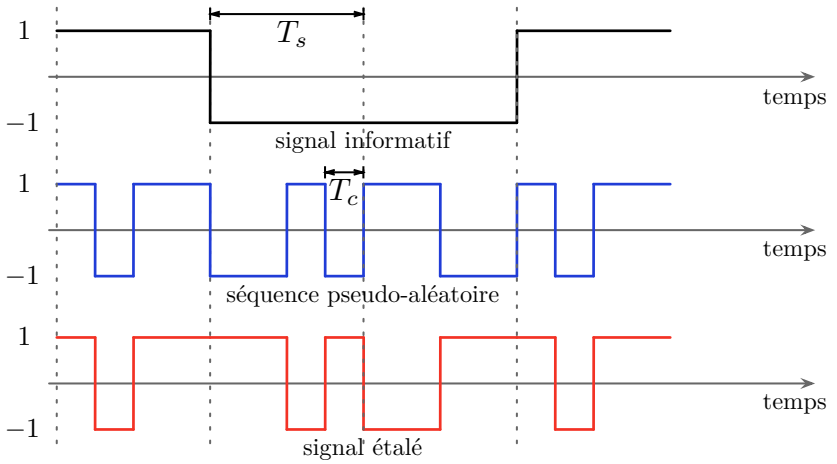


FIGURE 3.6: La technique d'étalement de spectre par séquence directe. La période du *chip* (T_c) est beaucoup plus faible que la période du symbole (T_s).

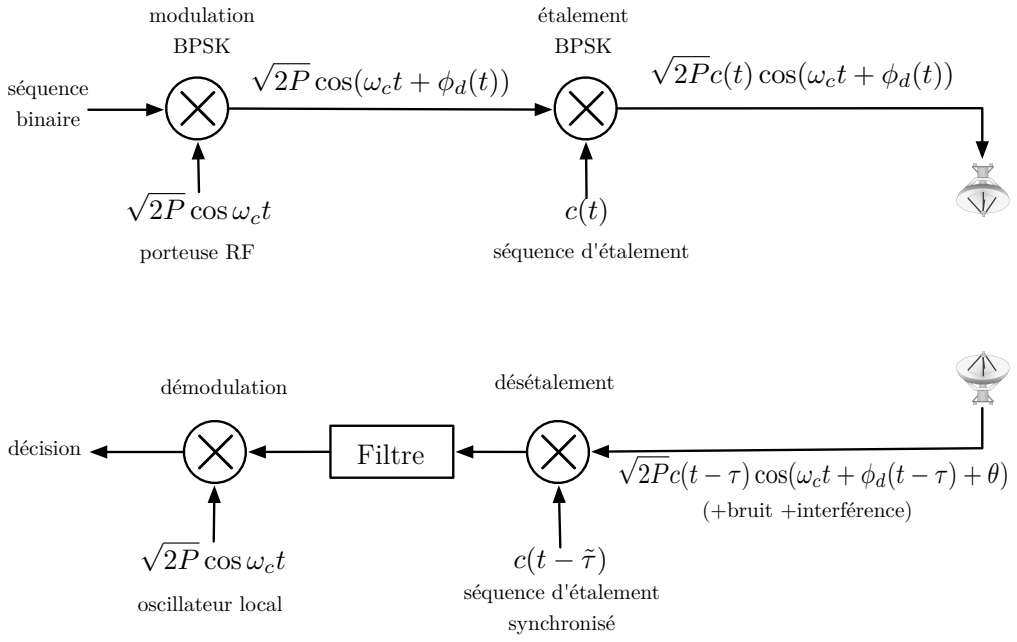


FIGURE 3.7: Principe d'un système de transmission utilisant la technique DSSS. Le signal est modulé en BPSK et l'étalement de spectre est fait avec une séquence pseudo–aléatoire binaire.

Le signal après désétalement est égal à

$$\sqrt{2P}c(t-\tilde{\tau})c(t-\tau)\cos(\omega_c t+\phi_d(t-\tau)+\theta) \tag{3.19}$$

où $\tilde{\tau}$ est le retard estimé. Si l'estimation est parfaite, i.e. $\tilde{\tau}=\tau$, le signal BPSK est retrouvé

$$\sqrt{2P}\cos(\omega_c t+\phi_d(t-\tau)+\theta) \tag{3.20}$$

car $c(t-\tilde{\tau})c(t-\tau)=1$. L'estimation de ce retard est réalisée dans le récepteur en râteau présenté brièvement dans la partie suivante.

3.2.2 Récepteur en râteau *(Rake Receiver)*

Dans les transmissions sans fil, le canal n'est pas statique mais varie au cours du temps. Le modèle de canal le plus utilisé est celui de Rayleigh [138]. Dans ce modèle, l'amplitude du signal suit une distribution de Rayleigh et possède une certaine probabilité d'être au-dessous du seuil de détection. En conséquence, le transmetteur nécessite plus de puissance pour atteindre le même taux d'erreur binaire que celui obtenu dans un canal BBGA. Par exemple, un TEB $P_e=10^{-3}$ nécessite $E_b/N_o=10$ dB pour un canal BBGA mais 27 dB pour un canal Rayleigh. La distance est encore plus large pour des TEB plus faibles. Le récepteur en râteau est conçu pour contrer ce problème, en exploitant les trajets multiples.

Diversité

La diversité est le fait que le récepteur reçoit plusieurs fois un même signal. La diversité peut être obtenue par le récepteur utilisant plusieurs antennes suffisamment espacées, ou par le transmetteur envoyant différentes versions du signal. Ces différences sont dans la polarisation, dans la fréquence ou dans le temps. Les autres types de diversité correspondent aux trajets multiples liés à l'environnement. Le récepteur reçoit le même signal mais avec différentes atténuations et différents retards.

Chaque trajet suit une distribution de Rayleigh, la probabilité que tous les trajets soient fortement atténués en même temps est visiblement plus faible que dans le cas d'un seul trajet. Cette probabilité est encore plus faible quand le nombre de trajets augmente. Le but est donc de combiner les informations issues des différents trajets pour avoir une meilleure décision. Cela est faisable dans le CDMA car le spectre est assez large pour différencier les trajets [2].

Combinaison

Supposons la présence de N_p trajets et soient A_i, ϕ_i et σ_i, respectivement l'amplitude du signal, la phase du signal et l'écart-type du bruit, avec $i=1,2,...,N_p$. Une combinaison linéaire optimale $\sum_{i=1}^{N_p} w_i u_i$ est recherchée de manière à maximiser le rapport signal sur bruit

2. Considérons la fréquence du *chip* à 3,84 Mcps (million de *chips* par seconde) de la norme WCDMA, la durée de *chip* est $0,26$ μs. Ceci est équivalent à une distance minimale de 78 m entre les trajets.

θ^2. Ce terme de RSB est calculé à l'aide de

$$\theta^2 = \frac{\left|\sum_{i=1}^{N_p} w_i A_i e^{j\phi_i}\right|^2}{\sum_{i=1}^{N_p} w_i^2 \sigma_i^2}. \tag{3.21}$$

L'inégalité de Schwarz permet d'obtenir

$$\theta^2 \leq \frac{\left(\sum_{i=1}^{N_p} w_i^2 \sigma_i^2\right)\left(\sum_{i=1}^{N_p} \frac{A_i^2}{\sigma_i^2}\right)}{\sum_{i=1}^{N_p} w_i^2 \sigma_i^2} = \sum_{i=1}^{N_p} \frac{A_i^2}{\sigma_i^2} \tag{3.22}$$

qui devient une égalité lorsque

$$w_i = w_i^* = k \frac{A_i}{\sigma_i^2} \exp(-j\phi_i) \tag{3.23}$$

avec k une constante. La détermination en parallèle des termes A_i, ϕ_i et σ_i pour calculer la valeur optimale w_i^* étant difficile, il existe différentes techniques de combinaison. Dans le récepteur WCDMA étudié par la suite, la technique de combinaison à ratio maximal (*Maximum Ratio Combination* - MRC) est utilisée. Cette technique consiste à supposer que les puissances du bruit associées aux trajets sont les mêmes. Ainsi, l'équation (3.23) devient $w_i = A_i e^{-j\phi_i}$. Cette valeur est estimée grâce à une séquence d'apprentissage introduite dans les trames DPCCH.

Récepteur en râteau

Dans cette partie, un récepteur en râteau de type MRC est considéré. Son principe et les techniques d'adaptation sont présentés. Le récepteur en râteau sera mieux détaillé dans la section 3.3.

Un récepteur en râteau de type MRC composé de n branches (ou *fingers*) est présenté dans la figure 3.8. Le nombre de branches dépend de la norme, et correspond aussi à un compromis entre les performances du système et la complexité du récepteur. Une branche du récepteur en râteau se compose de trois modules.

- Estimation de canal : l'amplitude complexe $\hat{\alpha}_i(t)$ du $i^{ème}$ trajet est estimée grâce à une séquence de bits pilotes.
- Décodage : le signal reçu est *désétalé*, il est multiplié par des versions retardées conjuguées du code d'étalement. Les délais sont estimés par un module de recherche de trajet, mais seuls les n plus importants trajets sont considérés. Ensuite, le signal désétalé est intégré sur une durée de symbole pour retrouver le signal avant l'étalement.
- Synchronisation fine : un système basé sur une boucle à verrouillage de délais (DLL, pour *Delay-Locked Loop*), attaché à chaque trajet, permet d'affiner le retard trouvé par le module de recherche de trajet.

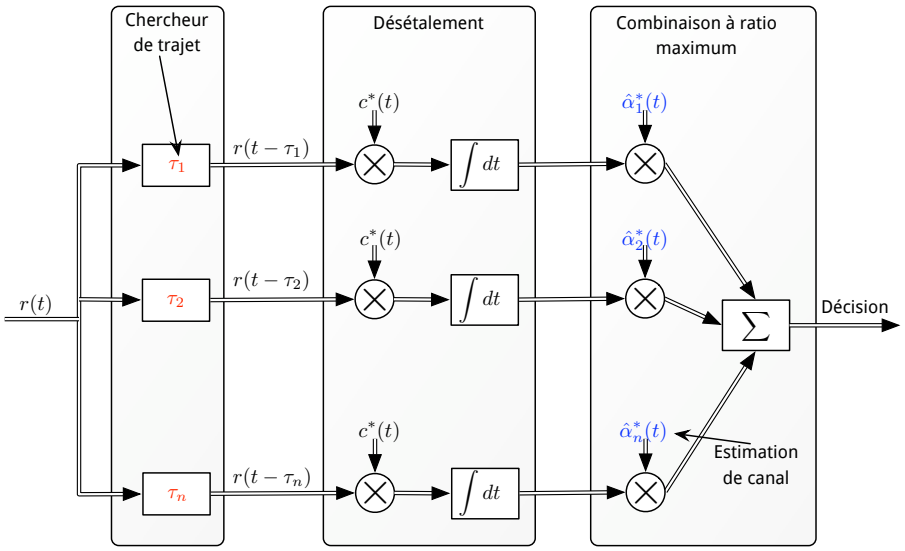


FIGURE 3.8: Récepteur en râteau de type MRC. τ_i vient du module de recherche de trajet et $\hat{\alpha}_i(t)$ vient de l'estimateur de canal.

Les sorties des branches du récepteur en râteau sont combinées (combinaison de type MRC). La sortie associée à chaque trajet est multipliée par le conjugué de l'estimation de l'amplitude complexe du canal, puis elles sont agrégées. Le résultat est ensuite transmis à l'élément de décision pour décider du symbole transmis.

Le signal reçu $r(t)$ est la somme des différents trajets $r_i(t)$, du bruit et des interférences

$$\begin{aligned} r(t) &= \sum_{i=1}^{N_p} (r_i(t) + b_{in}(i,t)) \\ &= \sum_{i=1}^{N_p} (\alpha_i(t) s(t-\tau_i) + b_{in}(i,t)) \qquad (3.24) \\ &= \sum_{i=1}^{N_p} (\alpha_i(t) d(t-\tau_i) c(t-\tau_i) + b_{in}(i,t)) \qquad (3.25) \end{aligned}$$

où $d(t)$ est le symbole d'information, $s(t)$ le signal étalé, $b_{in}(i,t)$ représente le terme incluant le bruit et les interférences associés au trajet i. Dans le processus de *désétalement*, dans chaque *branche*, le signal reçu est multiplié par une version décalée du conjugué du code d'étalement, ce qui donne

$$\begin{aligned} z_k(t) &= r(t) c^*(t-\tau_k^*) \\ &= \sum_{i=1}^{N_p} \alpha_i(t) d(t-\tau_i) c(t-\tau_i) c^*(t-\tau_k^*) + c^*(t-\tau_k^*) \sum_{i=1}^{N_p} b_{in}(i,t) \qquad (3.26) \\ &= \alpha_k(t) d(t-\tau_k) c(t-\tau_k) c^*(t-\tau_k^*) \\ &\quad + \sum_{i \neq k}^{N_p} \alpha_i(t) d(t-\tau_i) c(t-\tau_i) c^*(t-\tau_k^*) + c^*(t-\tau_k^*) \sum_{i=1}^{N_p} b_{in}(i,t) \qquad (3.27) \end{aligned}$$

avec τ_k^* le délai estimé du trajet k. Si $\tau_k = \tau_k^*$, le premier terme du membre de droite de l'équation (3.27) devient $\alpha_k(t)d(t-\tau_k)$. L'intégration de l'équation (3.27) sur une durée de symbole T_s conduit à

$$\begin{aligned} z_k(n) &= \alpha_k(n)N_c d(n-n_k) \\ &+ \sum_{i\neq k}^{N_p}\left(\alpha_i(n)d(n-n_k)\sum_{j=1}^{N_c} c(N_c(n-1)+j-n_i)c^*(N_c(n-1)+j-n_k)\right) \\ &+ \sum_{i=1}^{N_p}\sum_{j=1}^{N_c} c^*(N_c(n-1)+j-n_k^*)b_{in}(i,N_c(n-1)+j) \end{aligned} \tag{3.28}$$

Les deux derniers termes de l'expression (3.28) sont atténués car ils correspondent au produit de deux séquences aléatoires différentes. Particulièrement, lorsque les codes d'étalement sont orthogonaux, le deuxième terme est égal à zéro et le premier terme $\alpha_k(n)N_c d(n-n_k)$ est donc prépondérant. Cette partie du signal de chaque branche est transmise au module de combinaison à ratio maximal (MRC) puis ensuite à l'élément de décision.

3.2.3 Estimation de la dynamique

La première étape du processus de conversion en virgule fixe correspond à la détermination de la largeur de la partie entière. Il est nécessaire de déterminer la dynamique de chaque donnée dans cette étape. Une approche analytique reposant sur l'arithmétique d'intervalle [78] est utilisée pour estimer la dynamique et garantir l'absence de débordement. Cependant, cette méthode surestime parfois la dynamique si les propriétés liées à l'application ne sont pas prises en compte.

Dans un système d'étalement de spectre par séquence directe, le rapport signal sur bruit est particulièrement faible. Avec plusieurs dizaines d'utilisateurs simultanés en communications, la puissance de bruit et des interférences est nettement plus élevée que celle du signal utile. Ainsi, la dynamique est principalement due au bruit et aux interférences. Dans le processus de *désétalement*, le signal est accumulé sur la longueur L de la séquence d'étalement. L'approche purement analytique va multiplier la dynamique de l'entrée par L. Mais en fait, ce processus accumule principalement la dynamique du signal utile, non pas du bruit et des interférences. A travers cette propriété, une approche est proposée afin de déterminer plus précisément les dynamiques. Avant le processus de *désétalement*/corrélation, l'ensemble du signal utile plus bruit est considéré. Après ce processus, seul le signal utile est pris en compte lors du calcul de la dynamique.

Dans (3.24), le signal reçu correspond à la somme du signal utile associé à chaque trajet et une source de bruit et d'interférences $b_{in}(t)$. Cette source peut être considérée comme gaussien avec une variance σ_{in}^2. Dans le cas d'une modulation QPSK, supposons que le signal étalé prenne des valeurs dans $\{\pm 1 \pm i\}$ et que RSB $= 1/\sigma_{in}^2$. Au récepteur, supposons l'absence d'autres sources de signal. L'amplitude des parties réelle et imaginaire de la partie utile du signal se situe dans l'intervalle $[-1, 1]$. Le bruit et les interférences étant considérés comme gaussien, $99,7$ % de ses valeurs se trouve dans l'intervalle $[-3\sigma_{in}, 3\sigma_{in}]$. Le signal reçu est donc situé avec une grande probabilité dans l'intervalle $[-1-3\sigma_{in}, 1+3\sigma_{in}]$. Avec cette hypothèse, la normalisation du signal reçu est faite en divisant par $1+3\sigma_{in}$ afin

d'avoir un signal global dans l'intervalle $[-1, 1]$. Ainsi, la dynamique de la partie utile du signal se retrouve dans l'intervalle $[-\frac{1}{1+3\sigma_{in}}, \frac{1}{1+3\sigma_{in}}]$.

La multiplication du signal reçu et du conjugué du code d'étalement n'augmente pas la dynamique dans le cas de la modulation BPSK et augmente la dynamique d'un facteur deux dans le cas d'une modulation QPSK (en raison de la multiplication complexe). Après l'accumulation sur N *chips*, la partie bruit et interférence est fortement atténuée. Dans l'équation (3.28), nous considérons que la dynamique $z_k(n)$ est égale à la dynamique de $N_c \cdot a_k(n)$, soit

$$|\Re(z_k(n))| = |\Im(z_k(n))| = \frac{2N_c}{1+3\sigma_{in}}. \tag{3.29}$$

$z_k(n)$ est divisé par N_c et multiplié par la valeur conjuguée de l'estimation de l'amplitude complexe du canal $\hat{\alpha}_k^*(t)$ pour la combinaison à ratio maximal. Dans le cas du WCDMA, cette valeur est le résultat des mêmes calculs, mais la séquence d'apprentissage est utilisée à la place du signal informatif du canal DPDCH.

3.2.4 Évaluation de précision

Modélisation du bruit

Le bruit de quantification peut être modélisé par une somme de différentes sources de bruit se propageant dans tout le système. Cette somme peut être considérée comme une source de bruit unique $\widetilde{e_q}$ à la sortie du système. Dans [98], cette source de bruit $\widetilde{e_q}$ est validée pour un large éventail d'applications comme la somme d'un bruit uniforme et d'un bruit gaussien selon

$$\widetilde{e_q} = \upsilon(\beta \times e_u + (1-\beta) \times e_n) \tag{3.30}$$

où e_u et e_n sont respectivement un bruit uniformément distribué et un bruit gaussien ayant une variance unitaire. Le terme υ contrôle la variance (puissance) du bruit global et $\beta \in [0, 1]$ correspond à un poids permettant la combinaison des deux modèles. Si une source de bruit domine les autres bruits de quantification, le bruit à la sortie possède une distribution uniforme et $\beta \approx 1$. A l'autre extrémité, où chaque source de bruit contribue de la même manière, $\beta \approx 0$. Ce modèle est valable pour tous les systèmes basés sur les opérations arithmétiques et l'utilisation du mode de quantification par arrondi.

Évaluation de précision

Pour calculer l'expression de la puissance du bruit de quantification en sortie, la technique présentée dans [97] est utilisée. Étant donné que le code et l'amplitude complexe du canal sont constants dans une même trame, le système est considéré linéaire et invariant dans le temps. Pour le choix du code et de l'amplitude complexe du canal, le pire cas conduisant à la puissance du bruit maximal de quantification est considéré. Le bruit de quantification de sortie est une somme pondérée de chaque source de bruit ayant une variance $\sigma_{e_i}^2$. La puissance du bruit de quantification de sortie P_b, ou encore sa variance σ_q^2 est donnée par [97]

$$P_b = \sigma_q^2 = \sum_i K_i \sigma_{e_i}^2 \tag{3.31}$$

avec K_i le gain entre la sortie et la source de bruit. La variance $\sigma_{e_i}^2$ de chaque source de bruit est obtenue à partir du pas de quantification q_i (le bit le moins significatif) de la source de bruit considérée par l'expression

$$\sigma_{e_i}^2 = \frac{q_i^2}{12}. \tag{3.32}$$

Détermination de la contrainte de précision

Dans cette partie, l'expression du taux d'erreur binaire TEB(σ_q^2, RSB) en fonction de la puissance de bruit de quantification à la sortie et du RSB est présentée. Cette expression permet de déterminer la puissance maximale du bruit de quantification satisfaisant le critère présenté à l'équation (2.2).

Dans un premier temps (1), le cas d'une distribution gaussienne pour le bruit de quantification de sortie est considéré ($\beta = 0$). À l'intérieur du système, les bruits de quantifications multiples $e_1, e_2, ..., e_K$ sont générés. En l'absence de source de bruit dominante, en raison du théorème central limite, la somme de ces bruits est alors considérée suivre une densité de probabilité gaussienne selon

$$f_q(x) = \frac{1}{\sigma_q\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_q^2}}. \tag{3.33}$$

Dans un récepteur WCDMA, le bruit thermique et l'interférence d'accès multiples (MAI) peuvent être modélisé par un bruit gaussien si le canal de transmission est assimilable à un bruit blanc gaussien additif et en l'absence de brouilleurs dominants [123]. Dans les autres cas, l'approximation gaussienne améliorée [106] (IGA) ou d'autres méthodes doivent être utilisées. Par souci de simplicité, le signal reçu est supposé avoir deux composantes correspondant au signal désiré et au bruit (intégrant les interférences) gaussien. Ainsi, le bruit à la sortie est la somme du bruit de quantification à la sortie e_q et du bruit du récepteur propagé n_{out}. L'expression de la densité de probabilité du bruit global $f_n(x)$ est donc

$$f_n(x) = \frac{1}{\sqrt{\sigma_{n_{out}}^2 + \sigma_q^2}\sqrt{2\pi}} \exp \frac{-x^2}{2(\sigma_{n_{out}}^2 + \sigma_q^2)} \tag{3.34}$$

tandis que sa fonction de répartition $F_n(x)$ vaut

$$F_n(x) = \frac{1}{2}\left(1 + \operatorname{erf} \frac{x}{\sqrt{\sigma_{n_{out}}^2 + \sigma_q^2}\sqrt{2}}\right). \tag{3.35}$$

La technologie WCDMA utilisant la modulation QPSK, le taux d'erreur binaire peut être déterminé par

$$\begin{aligned} \text{TEB} &= 1 - F_n(1) = \frac{1}{2} \operatorname{erfc} \frac{1}{\sqrt{\sigma_{n_{out}}^2 + \sigma_q^2}\sqrt{2}} \\ &= Q\left(\frac{1}{\sqrt{\sigma_{n_{out}}^2 + \sigma_q^2}}\right) \end{aligned} \tag{3.36}$$

.

À partir des expressions (2.2) et (3.36), la condition sur σ_q^2 est donc la suivante :

$$\sigma_q^2 \leq \frac{1}{2}\text{erfc}^{-1}\left((1+\epsilon)\,\text{erfc}\,\frac{1}{\sigma_{n_{out}}\sqrt{2}}\right)^{-2} - \sigma_{n_{out}}^2 \tag{3.37}$$

soit

$$\sigma_q^2 \leq Q^{-1}\left((1+\epsilon)Q\left(\sigma_{n_{out}}^{-1}\right)\right)^{-2} - \sigma_{n_{out}}^2. \tag{3.38}$$

Dans un second temps (2), le cas d'un bruit de quantification proche de la sortie dominant est considéré ($\beta = 1$). Dans ce cas, la distribution du bruit est uniforme et sa densité de probabilité $f_q(x)$ est alors

$$f_q(x) = \frac{1}{q}\mathbf{Id}_{[\frac{-q}{2},\frac{q}{2}]} \tag{3.39}$$

Ainsi, le bruit global incluant la source de bruit gaussien f_c, a pour densité de probabilité

$$f_n(x) = f_c * f_q(x) = \int_{-\infty}^{\infty} f_c(t) f_q(x-t)dt \tag{3.40}$$

$$= \frac{1}{2q}\left(\text{erf}\,\frac{x+\frac{q}{2}}{\sqrt{N_0}} - \text{erf}\,\frac{x-\frac{q}{2}}{\sqrt{N_0}}\right) \tag{3.41}$$

et pour fonction de répartition

$$F_n(x) = \int_{-\infty}^{x} \frac{1}{2q}(\text{erf}\,\frac{t+\frac{q}{2}}{\sqrt{N_0}} - \text{erf}\,\frac{t-\frac{q}{2}}{\sqrt{N_0}}). \tag{3.42}$$

Dans le cas d'une modulation QPSK, le taux d'erreur binaire peut être déterminé par

$$\text{TEB} = 1 - F_n(1) \tag{3.43}$$

De la même manière que précédemment, la précision peut être déduite à partir des expressions (2.2) et (3.43). L'absence d'expression mathématiquement simple nécessite de résoudre de problème numériquement.

3.2.5 Conclusion

Dans cette partie, la méthodologie mise en œuvre pour déterminer les spécifications virgule fixe dans le cadre des systèmes de transmission numérique a été détaillée. Un récepteur QPSK a été étudié pour connaître la relation entre le bruit de quantification et le taux d'erreur binaire, en fonction du bruit de canal. Nous avons également étudié les propriétés des système DS-CDMA pour mieux adapter les spécifications. La technologie d'étalement de spectre est prise en compte dans notre approche.

3.3 Adaptation du récepteur WCDMA

L'évolution et la diversité des services de données nécessitent un débit de plus en plus élevé. Par exemple, le service de voix du GSM requiert seulement 9,6 Kbit/s mais la transmission de vidéo et de données du EDGE et du WCDMA peut utiliser de quelques dizaines Kbit/s à quelques centaines de Kbit/s. La complexité de la partie bande de base de plus en plus augmente, et il est nécessaire de trouver une implantation de ces applications efficace d'un point de vue énergétique. Nous nous somme intéressés plus particulièrement à un récepteur 3G WCDMA.

3.3.1 Présentation du système WCDMA

Le WCDMA *(Wideband Code Division Multiple Access)* est une technique utilisée pour la téléphonie mobile de troisième génération. La norme UMTS est basée sur cette technique. Dans ces systèmes utilisant la technique CDMA, un récepteur en râteau *(Rake Receiver)* est utilisé pour contrer les effets d'atténuation à trajets multiples. Une branche *(finger)* est allouée à chaque trajet pour décoder le symbole associé à ceux-ci. Un élément important associé au récepteur en râteau est le *module de recherche du trajet (path searcher)*. Ce module détermine le retard de chaque trajet. Ensuite, l'estimation du retard est utilisée pour synchroniser le signal d'entrée avec le code généré dans le récepteur et obtenir ainsi une combinaison optimale de l'énergie reçue.

Dans la technologie WCDMA, deux couches de code d'étalement [145], un code de multiplexage et un code d'embrouillage sont utilisées. Le code de multiplexage C_{ch} est utilisé pour obtenir l'orthogonalité entre les canaux lorsque le décalage du temps est égal à 0. Les codes d'embrouillage utilisés en sens montant sont des codes de Gold S_G. Les données d'entrée $d(n)$ sont multipliées par des codes d'étalement, et l'expression du signal transmis $\mathrm{Tx}(n)$ est

$$\mathrm{Tx}(n) = d(n)C_{ch}(n)S_G(n). \tag{3.44}$$

Dans un canal de Rayleigh à trajets multiples, le signal global reçu $\mathrm{Rx}(n)$ est la somme des signaux élémentaires $\mathrm{Rx}_k(n)$ pour les différents trajets. Chaque signal élémentaire $\mathrm{Rx}_k(n)$ pour un trajet k vaut

$$\mathrm{Rx}_k(n) = a_k\mathrm{Tx}(n - n_k) + n_k(n) + i_k(n), \tag{3.45}$$

où $\tau_k = n_k T_c$ est le retard associé au $k^{\text{ème}}$ trajet dans le canal, a_k est l'atténuation du trajet, $n_k(n)$ et $i_k(n)$ sont respectivement les bruits (blancs gaussiens additifs) et les interférences au récepteur.

Un doigt est composé de deux parties principales correspondant au module de décodage symbole et au module d'estimation de canal. Le graphe de flot de données d'un doigt est présenté dans la figure 3.10. L'amplitude complexe α_k du $k^{\text{ème}}$ trajet est estimée par la séquence pilote dans le trame de contrôle (DPCCH). Grâce à la multiplication complexe du signal reçu par le code Gold conjugué, l'opération de désembrouillage est effectuée. Ensuite, l'opération de désétalement transforme le signal reçu à large bande en un signal à bande étroite. Enfin, la distorsion de phase résultant du canal de transmission est supprimée.

Chaque doigt nécessite la connaissance du retard τ_k du $k^{\text{ème}}$ trajet. Tout d'abord, l'estimation grossière du temps de retard est réalisée avec le module de recherche de trajet.

Ensuite, la synchronisation fine du signal reçu est faite avec une DLL. Le graphe de flot de données du module de recherche de trajets est présenté dans la figure 3.17. Le signal reçu est multiplié par différentes versions décalées des codes d'étalement. Un seuil est calculé sur la puissance moyenne, puis les pics sont déterminés. Chaque pic donne potentiellement un trajet avec le retard correspondant.

Paramètres de simulation Dans notre expérimentations, un modèle de canal de Rayleigh respectant le troisième cas de la norme 3GPP [1], sans effet de Doppler est utilisé. Ceci correspond à une atténuation multi-trajet avec quatre composantes (gain, retard) : (0 dB, 0 ns) ; (–3 dB , 261 ns) ; (–6 dB, 521 ns) ; (–9 dB, 781 ns). Chaque utilisateur utilisateur occupe un seul canal DPDCH avec un facteur d'étalement de spectre SF = 16. Ceci correspond au canal physique dédié aux données de la norme WCDMA dans le cadre de communication sans fil UMTS/3G à la vitesse de 240 Kbps, sans codage de canal.

Distribution de la puissance sur un récepteur

La puissance d'émission du récepteur mobile varie sur un large intervalle. La distribution de la puissance est illustrée dans la figure 3.9. La valeur maximale de la puissance d'émission est de 20 dBm et en pratique la puissance d'émission est considérée être entre 10 mW et 100 mW. Nous déduisons que le rapport de signal sur bruit et interférence varie également sur un grand intervalle. Dans nos expérimentations, la largeur de cet intervalle est fixée à 20 dB.

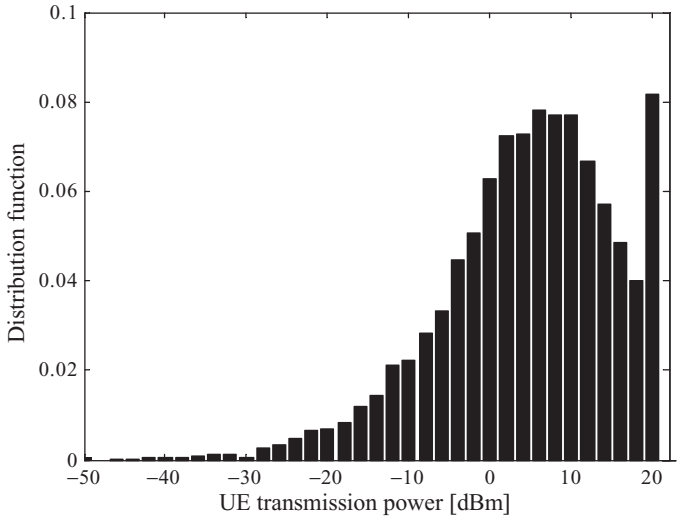


FIGURE 3.9: Distribution de la puissance d'émission du récepteur mobile à 64 kbps avec une probabilité de coupure de 5% selon [73].

3.3.2 Décodeur de symbole

Au récepteur, le délai τ_k de chaque trajet est tout d'abord estimé grossièrement par le module de recherche de trajets puis affiné par le système de synchronisation fine (DLL). Une DLL est associée à chaque trajet. Les symboles reçus $r_k(n)$ sont décodés par un

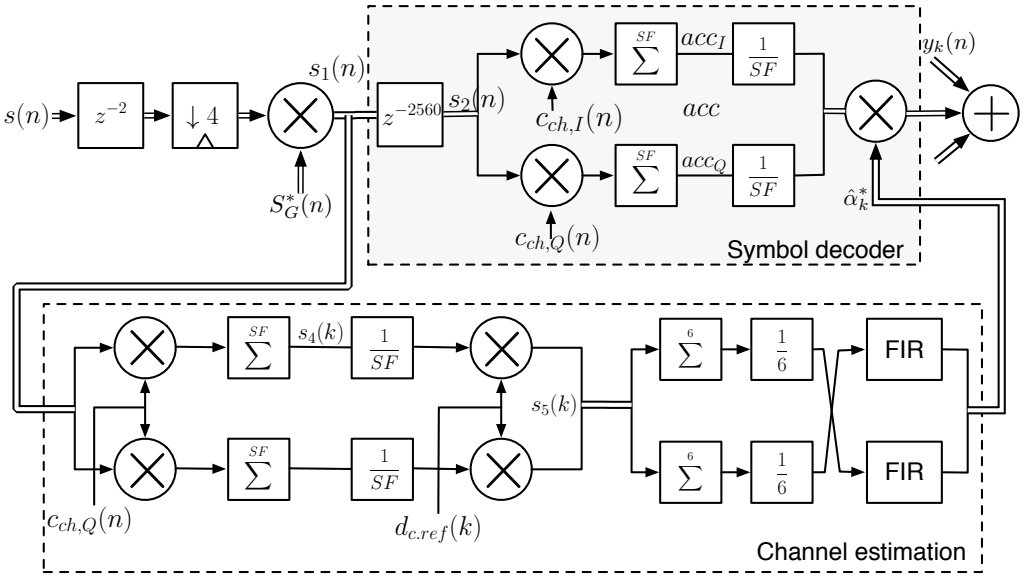


FIGURE 3.10: Graphe flot de données d'un décodeur de symboles dans un récepteur WCDMA.

récepteur en râteau composé de plusieurs branches. Chaque trajet k est traité par une branche du récepteur en râteau et est *désétalé* en multipliant le signal reçu par les valeurs conjuguées des codes d'étalement.

Le signal reçu r_k est accumulé sur une durée de symbole (SF chips) et sous échantillonné d'un facteur SF (facteur d'étalement). En raison des propriétés du code, le terme de bruit est négligeable par rapport au terme correspondant au signal utile dont le processus de corrélation avec les codes C_{ch} et S_G^* permet d'amplifier le signal utile d'un facteur SF. Le signal desétalé y_k de la branche k est démodulé grâce à une somme pondérée, de type MRC, des composantes.

Estimation de la dynamique

Le graphe flot d'un doigt du récepteur en râteau est présenté dans la figure 3.10. Dans le doigt, la corrélation entre le signal d'entrée et les codes est utilisée pour amplifier le signal utile et ainsi détecter le symbole transmis. Le processus de corrélation augmente la dynamique du signal utile, mais pas celle du bruit. Une approche est proposée afin de déterminer plus exactement la dynamique des données. Avant le processus de corrélation, le signal utile et le bruit sont considérés. Après ce processus, seul le signal utile est pris en compte lors du calcul de la dynamique.

D'après (3.45), le signal reçu est

$$\mathrm{Rx}(n) = \sum_{k=1}^{N_p} \mathrm{Rx}_k(n) = \sum_{k=1}^{N_p} a_k \mathrm{Tx}(n - n_k) + ni_k(n) \tag{3.46}$$

où ni_k est le terme représentant les interférences et le bruit, qui peut être considéré gaussien avec une variance de σ^2. Supposant que l'utilisateur possède un seul canal DPDCH, d'après (3.44), $d(n)C_{ch}(n)$ prend ses valeurs dans $\{\pm 1 \pm i\}$. Ainsi $\mathrm{Tx} = d(n)C_{ch}(n)S_G(n) \in$

$\{\pm 2, \pm 2i\}$. Dans nos simulations, $\mathrm{Tx}(n)$ est normalisé dans $\{\pm 1, \pm i\}$, ainsi sa puissance est égale à 1. Le rapport signal à bruit, par définition, est égal à $1/\sigma^2$.

L'augmentation du nombre des utilisateurs dans une cellule augmente les interférences. Ces interférences sont traitées comme des bruits et donc diminuent le RSB. Pour tenir compte des différents cas, correspondant à un petit nombre de communications dans une cellule avec de bonnes conditions de transmission et un grand nombre d'utilisateurs avec de mauvaises conditions de transmission, une grande variété de RSB est considérée (de 0 dB à 25 dB).

En utilisant le raisonnement de 3.2.3, l'entrée $s(n)$ comprenant le signal et le bruit est normalisée pour avoir l'amplitude des deux parties réelles et imaginaires dans l'intervalle $[-1, 1]$. Un bruit gaussien de variance σ^2 possède 99,7% de ses valeurs dans $[-3\sigma, 3\sigma]$. En supposant que les parties réelles et imaginaires de $\sum a_k \mathrm{Tx}(n - n_k)$ sont dans $[-1, 1]$, l'entrée est considérée être dans dans $[-1-3\sigma_{n_{ni}}, 1+3\sigma_{n_{ni}}]$. Le processus de normalisation est donc mis en œuvre en divisant l'entrée par $1 + 3\sigma_{n_{ni}}$.

En propageant la dynamique de l'entrée dans le graphe flot, les résultats obtenus sont les suivants. La dynamique de la sortie de l'accumulateur acc_I avant la normalisation est égale à

$$\max(|acc_I|) = \frac{4.SF}{1 + 3\sigma_{n_{ni}}}. \tag{3.47}$$

La dynamique de α_i correspondant à la sortie du module d'estimation de canal est égale à

$$\max(|\alpha_i|) = \frac{1}{1 + 3\sigma_{n_{ni}}}. \tag{3.48}$$

Enfin, la dynamique de y_k correspondant au module d'estimation de symbole est égale à

$$\max(|y_k|) = \frac{4}{(1 + 3\sigma_{n_{ni}})^2}. \tag{3.49}$$

Les dynamiques ont été estimées avec notre approche analytique pour différentes valeurs de RSB et comparées aux résultats obtenus à partir de simulations. Les résultats sont présentés dans la figure 3.11 pour la sortie acc_I de l'accumulateur et pour la sortie y_k de décodage symbole. La différence entre les estimations et les résultats simulés est de un à deux bits, selon les valeurs du RSB. Néanmoins, l'évolution de la dynamique en fonction du rapport signal sur bruit est identique pour l'estimation analytique et pour le résultat de la simulation. Cela confirme la validité de notre approche d'estimation de la dynamique dans le récepteur WCDMA. La différence entre les deux courbes s'explique par le modèle de canal plus réaliste utilisé dans la simulation. Si un modèle de canal mono-trajet BBGA est utilisé, la différence devient inférieure à 1 bit. De plus, les estimations analytiques sont connues pour être plus pessimistes, mais aussi beaucoup plus rapides à obtenir. Pour la sortie de l'accumulateur, entre l'estimation et la simulation, il y a une différence de 3 bits entre 0 dB et 25 dB. Pour la sortie du *finger*, il y a une différence de 4 bits entre 0 dB et 15 dB et de 6 bits entre 0 dB et 25 dB. Ces résultats montrent la possibilité d'adapter la partie entière des données en fonction du rapport signal sur bruit à l'entrée du récepteur.

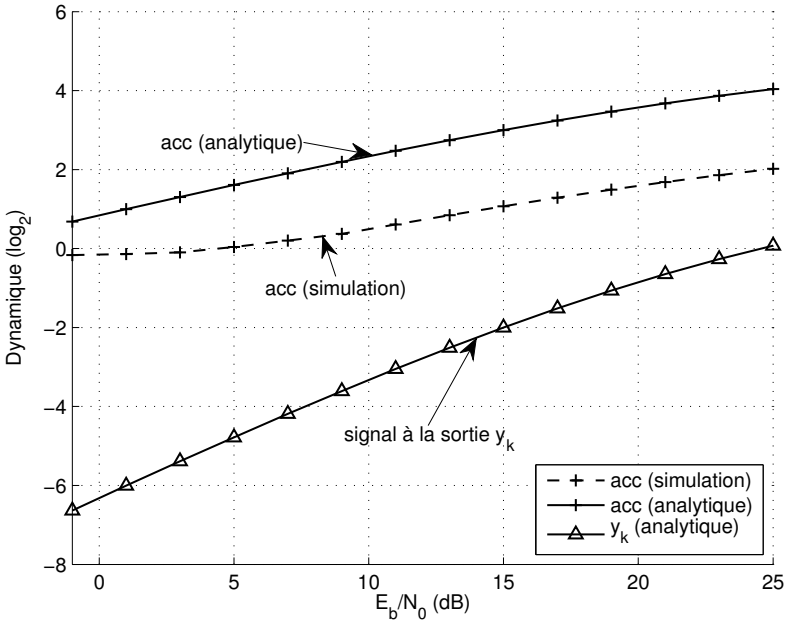


FIGURE 3.11: Valeur estimée et valeur réelle de la dynamique dans un décodeur pour différents RSB.

Optimisation de la largeur des données

De même que (3.14), le bruit de quantification total dans un décodeur de symbole peut être présenté par la somme des bruits de quantification à chaque étape des calculs. Supposons que le nombre de bits pour la partie fractionnaire dans chaque quantification est égal à b, les calculs montrent que

$$\sigma_{Q,I}^2 = \frac{2^{-2b}}{12} \times 9 \cdot \mathrm{SF} \tag{3.50}$$

$$\sigma_{Q,Q}^2 = \frac{2^{-2b}}{12} \times 9 \cdot 256 \tag{3.51}$$

avec $\sigma_{Q,I}^2$ et $\sigma_{Q,Q}^2$ est respectivement la puissance de bruit de quantification sur les voie I et Q.

La puissance de bruit après la normalisation est égale à $\sigma'^2 = \sigma^2 \times \frac{1}{(1+3\sigma)^2}$. La normalisation du signal a un impact sur le bruit. Il est montré dans la figure 3.12 qu'au moins 10 bits pour la partie fractionnaire sont nécessaires pour être proche de la précision infinie quand E_b/N_0 varie entre 0 dB et 10 dB.

La contrainte de précision est déterminée pour différents rapports signal sur bruit. Les résultats sont présentés dans la figure 3.13. Les courbes P_{y_k} et $P_{n_{out}}$ correspondent respectivement à la puissance du signal désiré (symbole) y_k et à la puissance du bruit en sortie n_{out}. La différence entre P_{y_k} et $P_{n_{out}}$ correspond au rapport signal sur bruit à la sortie. La différence entre P_{y_k} et P_{e_q} correspond au rapport signal sur bruit de quantification à la sortie (RSBQ). Les résultats montrent que le RSBQ doit être augmenté afin de maintenir le TEB lorsque le RSB croît. Dans un tel cas, une plus grande précision est nécessaire pour réduire des erreurs de décision liées à l'arithmétique en virgule fixe.

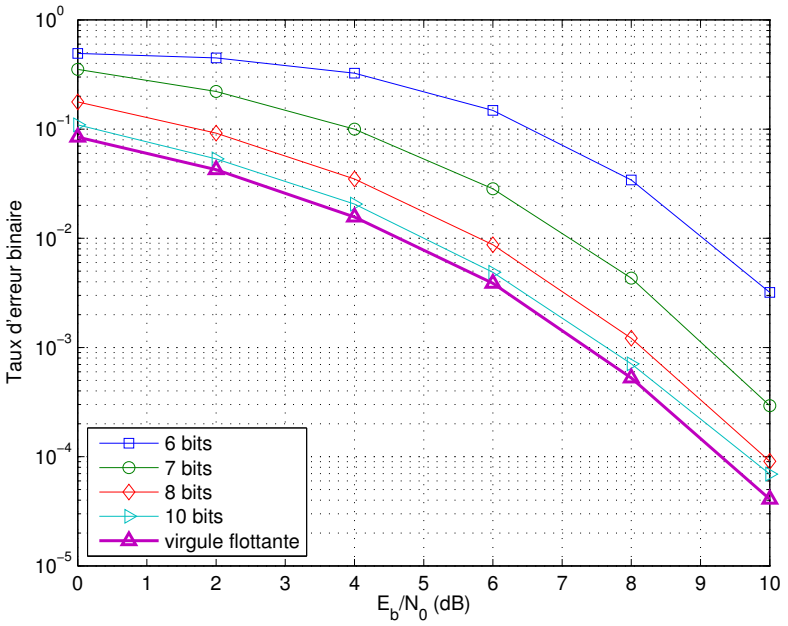


FIGURE 3.12: Résultats de simulation du TEB dans les conditions de canal différentes pour le décodeur de symbole.

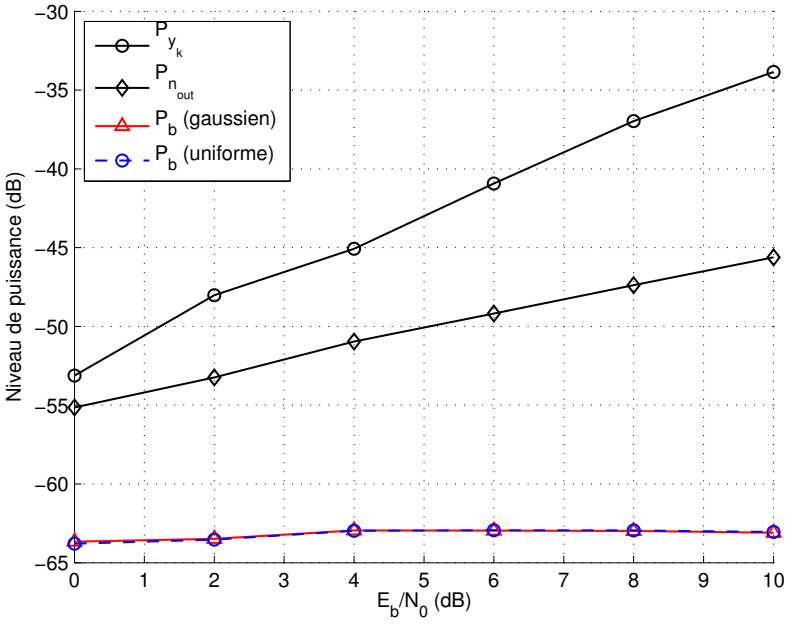


FIGURE 3.13: Puissance du signal et du bruit en fonction du RSB.

Le processus d'optimisation présenté à l'équation (4.2) est réalisé avec la contrainte de précision définie dans l'équation (3.37). Les algorithmes d'optimisations (présentés dans la deuxième partie de cette thèse) sont utilisés pour trouver les largeurs optimales pour chaque donnée. Dans un premier temps, pour analyser le potentiel de notre approche en terme d'économie d'énergie, une architecture ayant une granularité fine en termes de largeur de données est considérée. Une librairie d'opérateurs arithmétique ciblant des ASIC pour une technologie 180 nm est utilisée. L'optimisation des largeurs obtenue pour différentes valeurs de RSB est présentée dans le tableau 3.1. Les résultats montrent que la largeur optimisée varie en fonction du RSB. Entre, 0 dB et 20 dB la largeur de *acc* et y_k augmentent respectivement de 80% et 66%.

TABLE 3.1: Largeur optimale d'un récepteur en râteau obtenue pour différents rapports signal sur bruit.

E_b/N_0 (dB)	1	3	5	7	9	11	13	15	17	19	21
acc	5	6	6	6	7	7	7	7	9	9	9
y_k	6	7	7	7	8	9	9	9	9	9	10

La consommation d'énergie associée à chaque valeur de RSB est normalisée et présentée dans la figure 3.14. Les résultats d'optimisation montrent que, pour un RSB variant de 0 dB à 20 dB, nous pouvons potentiellement économiser jusqu'à 50% de la consommation d'énergie si la spécification en virgule fixe est adaptée en fonction du RSB.

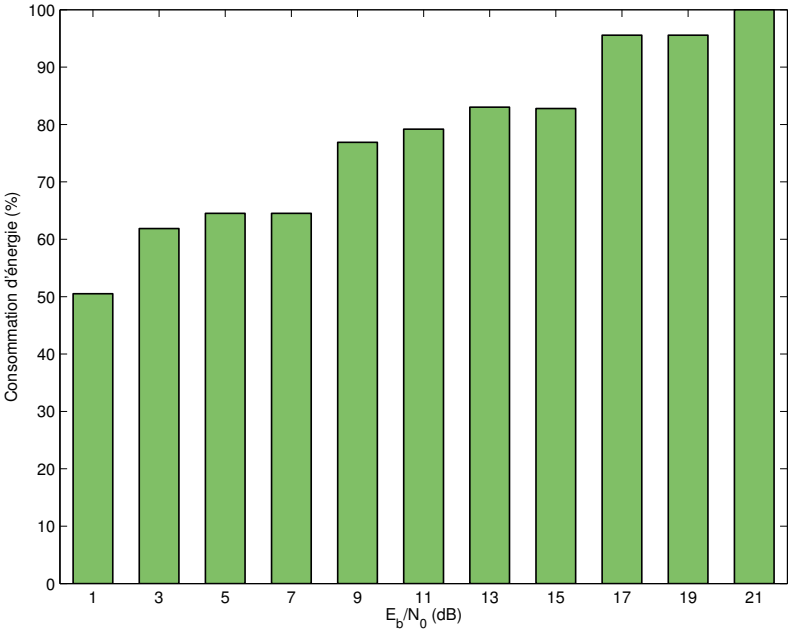


FIGURE 3.14: Consommation d'énergie normalisée du décodeur sur une cible ASIC.

Cette optimisation est expérimentée également avec une architecture intégrant des opérateurs SWP supportant un nombre de largeurs de données limité. Les largeurs supportés par l'opérateur SWP 40 bits utilisé sont de 5, 6, 8, 10 et 13 bits. Le résultat est présenté dans la figure 3.15. Nous observons la même tendance que précédemment. Sur le même

intervalle de 20 dB, la différence de consommation d'énergie est d'environ 50%.

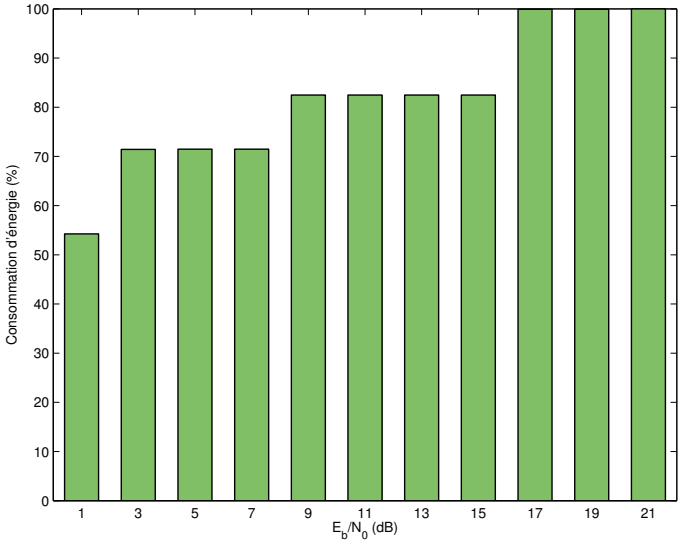


FIGURE 3.15: Consommation d'énergie normalisée du module de décodage dans le cas d'une implantation avec des opérateurs SWP.

Analyse en fonction du facteur d'étalement

Les expérimentations présentées ci dessus, pour la même cible ASIC 180 nm et une granularité fine, ont été réalisées pour différents facteurs d'étalement SF. La figure 3.16 présente la variation de la consommation d'énergie du décodeur pour des facteurs d'étalement de 16, 64 et 256. Un même intervalle du RSB a été utilisé. Cependant, le changement du facteur d'étalement entraîne le changement de E_b/N_0. Chaque facteur d'étalement conduisant à des paramètres différents pour le traitement, les configurations de format de données sont différentes pour chaque facteur d'étalement en particulier liées à des dynamiques des signaux différents. Les résultats montrent que la consommation d'énergie est différente en fonction du facteur d'étalement. Ainsi, l'approche d'adaptation dynamique de la précision permet de choisir pour chaque valeur de paramètre du système, la configuration virgule fixe adaptée.

3.3.3 Module de recherche de trajets

Dans cette section, le module de recherche de trajet est étudié. Celui-ci est basé sur un algorithme utilisant le profil de puissance retardée (*Power Delay Profile* – PDP). Il analyse dans une fenêtre temporelle de largeur multiple d'un chip, la corrélation entre le signal d'entrée et le code généré à l'intérieur du récepteur. Ce module réalise la synchronisation gros grain du retard des trajets, puis la synchronisation fine sera réalisée par la boucle à verrouillage de délai associée à chaque branche du récepteur en râteau.

Le graphe flot de données est présenté dans la figure 3.17. Premièrement, le module calcule la corrélation entre le signal d'entrée et le code associé au canal de contrôle (DPCCH). Ensuite, le module au carré de la corrélation est calculé et cette valeur est comparée à un

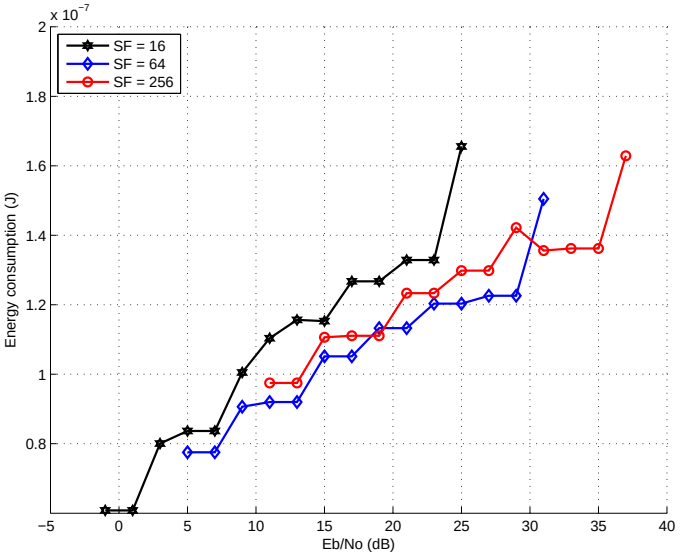


FIGURE 3.16: Énergie consommée pour le module de décodage pour différents facteurs d'étalement (SF)

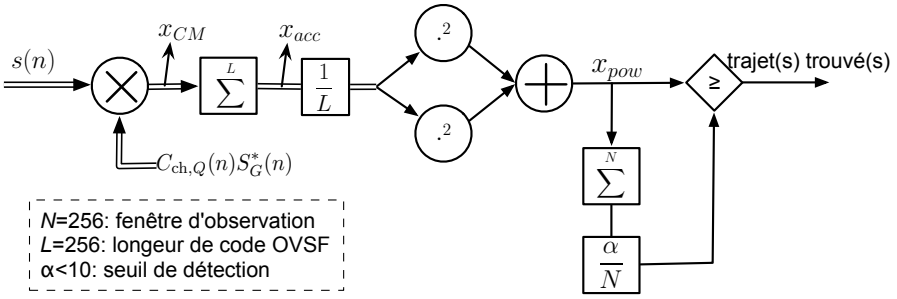


FIGURE 3.17: Graphe flot de données d'un module de recherche de trajet PDP (*Power Delay Profile*)

seuil adaptatif $t(l)$. Un trajet est détecté si cette valeur est supérieure au seuil. Le seuil adaptatif est proportionnel à la moyenne de toutes les valeurs de corrélation.

Estimation de la dynamique

L'approche présentée dans la partie 3.3.2 est utilisée pour estimer la dynamique du module de recherche de trajets. Le signal d'entrée Rx est normalisé dans l'intervalle $[-1, 1]$. Ce signal est multiplié par le code d'étalement complexe conjugué $C_{ch}S^*_G$. Les résultats sont situés dans l'intervalle $[-2, 2]$ pour chaque partie réelle et imaginaire. Pour l'accumulation sur L (la longueur du code OVSF) chips, seule la partie signal correspondant au symbole transmis est amplifiée de manière significative. Ainsi, l'expression de la dynamique est égale à

$$\max(|x_{acc}|) = \frac{2L}{1 + 3\sigma}. \tag{3.52}$$

La dynamique du signal x_{pow} correspondant au profil de puissance est égale à

$$\max(|x_{pow}|) = \frac{8}{(1+3\sigma)^2}. \tag{3.53}$$

La dynamique obtenue à l'aide des expressions 3.52 et 3.53 par simulation pour différentes valeurs de RSB est présentée dans la figure 3.18. La différence entre les résultats estimés et simulés est de 1 à 2 bits, mais les évolutions dans les deux cas sont identiques.

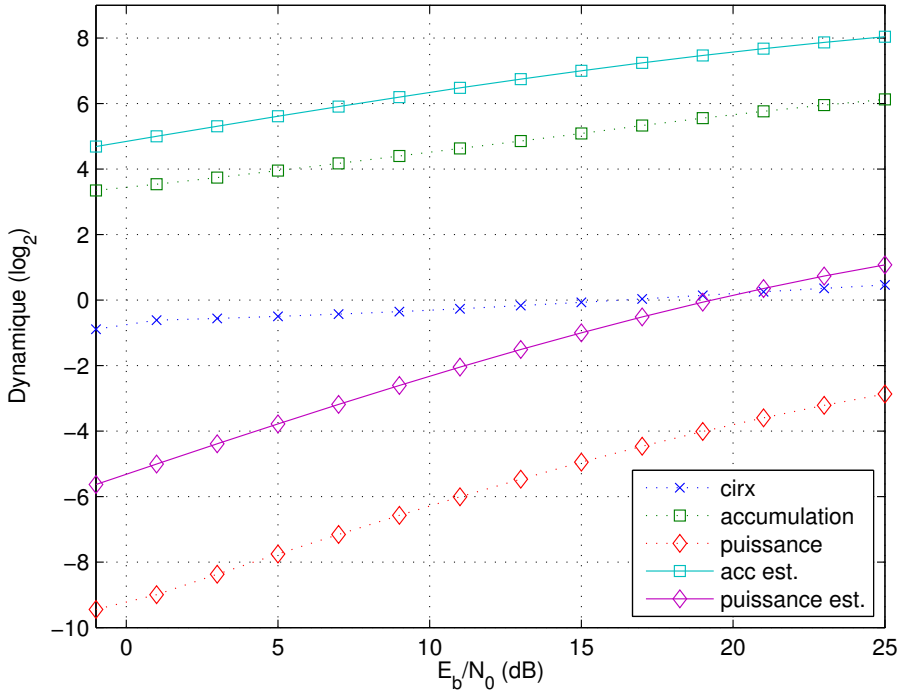


FIGURE 3.18: Dynamique au sein du module de recherche de trajets obtenue par simulation et par la méthode analytique.

Évaluation de la précision

Les critères de performance présentés dans la partie 2.3 ne peuvent pas être utilisés pour le module de recherche de trajets. Ce module est basé sur la théorie de la détection et les performances sont analysées à travers la probabilité de non-détection et la probabilité de fausse alarme.

Pour analyser les performances du module de recherche de trajets, le canal multi-trajets de Rayleigh est considéré. La sortie de ce module, avant la décision x_{pow}, est constituée de trois composantes correspondant au signal s_{pow}, au bruit du récepteur n_{pow} et au bruit de quantification à la sortie e_q. Par rapport au récepteur en râteau, la distribution du signal de sortie n'est pas régulière. Deux cas sont à considérer. En présence d'un trajet, la valeur de sortie dépend du module de l'amplitude complexe a_k associée au $k^{\text{ème}}$ trajet. En l'absence de trajet, les valeurs de sortie dépendent des propriétés du code. Dans ce dernier cas, la modélisation de la distribution du signal de sortie est complexe. Ainsi, la technique basée sur la simulation présentée dans [98] a été retenue pour déterminer la contrainte

de précision et une méthode de Monte Carlo est utilisée pour mesurer les probabilités de non-détection et de fausse alarme. Un modèle de canal de Rayleigh respectant le cas 3 de la norme 3GPP [1] sans effet de Doppler est utilisé, ce qui correspond à une atténuation des multi-trajets avec quatre composantes (gain, le retard) : (0 dB, 0 ns) ; (–3 dB, 261 ns) ; (–6 dB, 521 ns) ; (–9 dB, 781 ns). Les simulations sont effectuées avec 500 instances de ce canal pour chaque type de canal de Rayleigh.

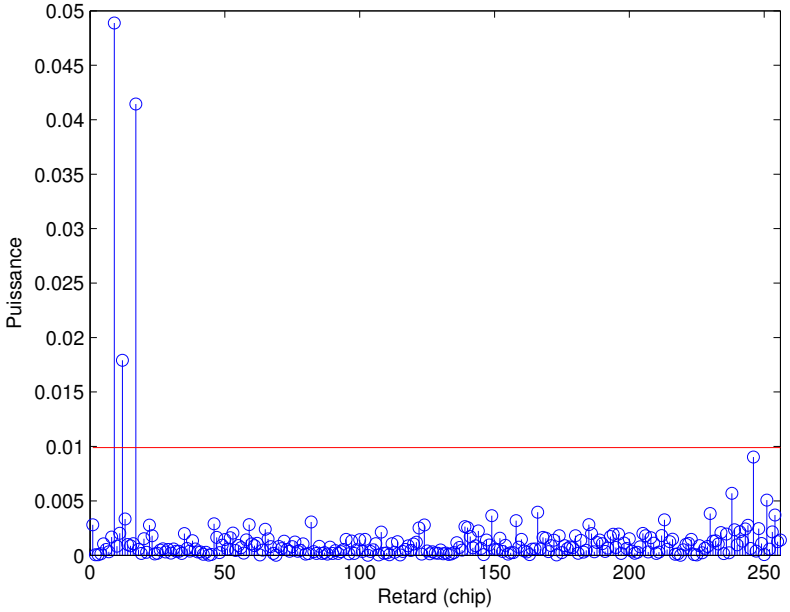


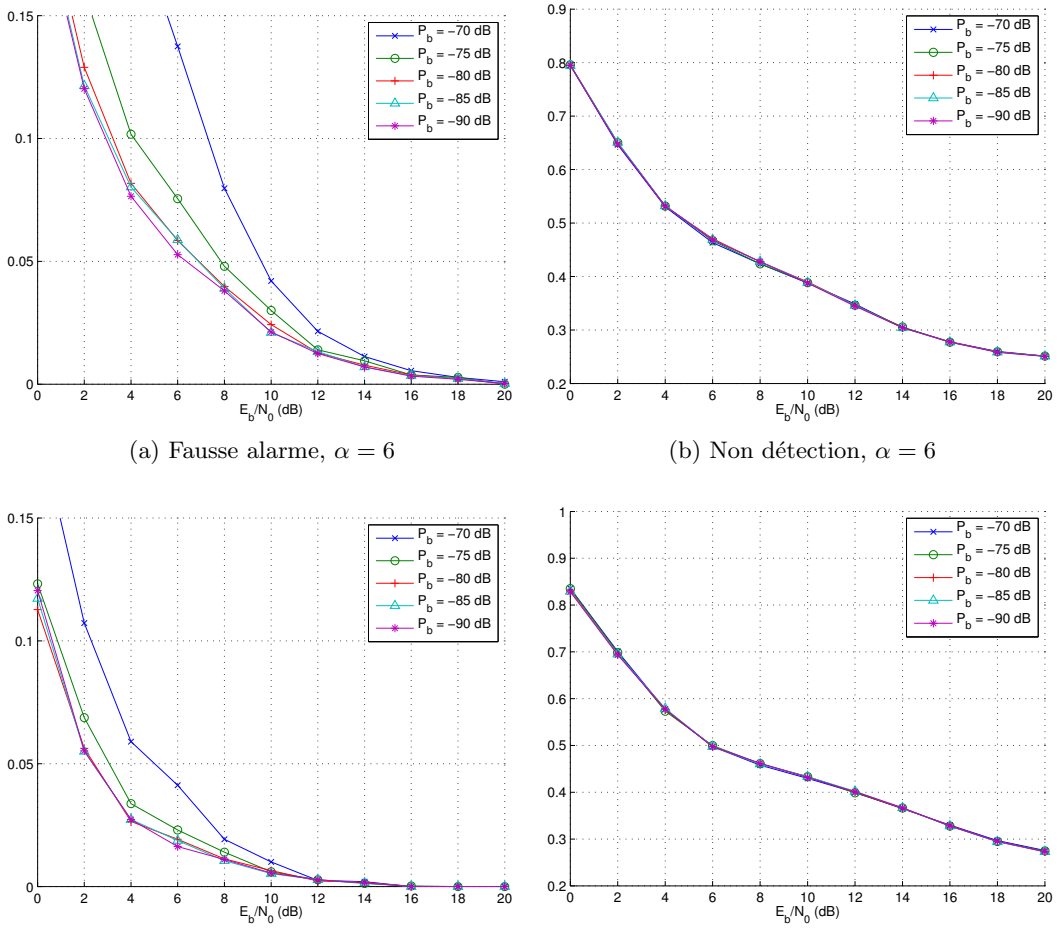
FIGURE 3.19: Une instance du profil de puissance pour un canal de Rayleigh de la norme 3GPP cas 3. SNR à 10 dB et α fixé à 7.

Dans le module de recherche de trajets, un paramètre important est le seuil de détection α. Un trajet est détecté si sa puissance est supérieure à $m_x.\alpha$ avec m_x la valeur moyenne de toutes les accumulations. Dans [34], une expression générale du seuil de détection est proposée :

$$\alpha = a + bK^c \tag{3.54}$$

avec K le nombre de trames utilisées et les termes a, b et c sont des paramètres du système. Dans [34] l'auteur propose les valeurs suivantes $a = 2, b = 4, c = -0,5$ pour la détermination de α. Ces valeurs conduisent à de très bons résultats sur plusieurs modèles de canal. Des valeurs α autour de 6 ont été testées. Le résultat présenté dans la figure 3.20 montre que la valeur $\alpha = 7$ réduit nettement le taux de fausse alarme en gardant le nombre de non-détections raisonnable. Cette valeur est alors utilisée dans nos expérimentations.

Afin de déterminer la puissance maximale du bruit de quantification, nous examinons la probabilité de fausse alarme et de non-détection pour différentes valeurs de la puissance du bruit de quantification : -60 dB à -90 dB. L'évolution en fonction du RSB de la probabilité de fausse alarme et de non-détection est présentée à la figure 3.20c, figure 3.20d et figure 3.21 pour différents niveaux du bruit de quantification de sortie. La probabilité de non-détection est moins sensible au bruit que celle liée aux fausses alarmes. Pour un même RSB, la probabilité de non-détection n'évolue pas et est très proche de celle obtenue en précision infinie. Par conséquent, la contrainte de précision est déterminée à partir des



(a) Fausse alarme, $\alpha = 6$

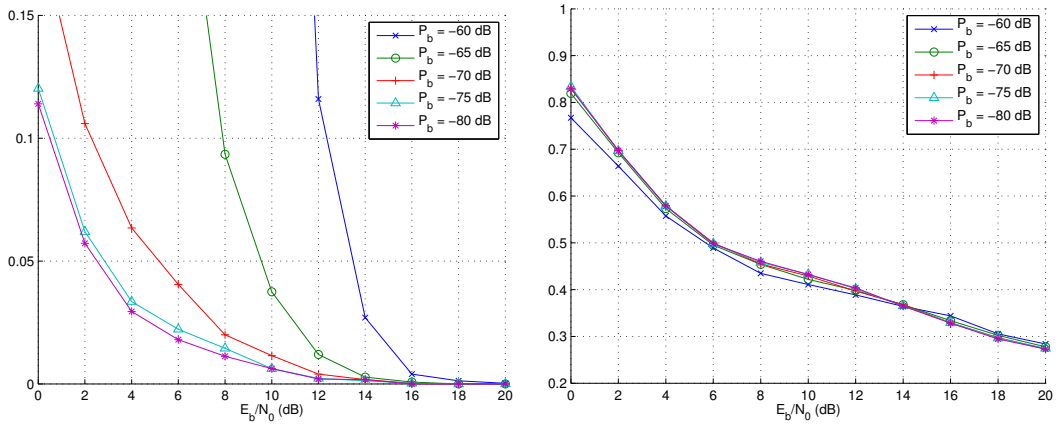
(b) Non détection, $\alpha = 6$

(c) Fausse alarme, $\alpha = 7$

(d) Non détection, $\alpha = 7$

FIGURE 3.20: Comparaison du taux de fausse alarme et taux de non-détection sur différents seuils : $\alpha = 6$ et $\alpha = 7$. Le niveau de bruit de quantification varie entre -70 dB à -90 dB.

critères sur les fausses alarmes. La puissance maximale du bruit de quantification de sortie est choisie, en fonction du RSB, de manière à ce que la probabilité de fausse alarme soit proche de celle obtenue en précision infinie.



(a) Taux de fausse alarme, $\alpha = 7$

(b) Taux de non-détection, $\alpha = 7$

FIGURE 3.21: Taux de FA et taux de MD avec $\alpha = 7$. Le niveau de bruit de quantification varie entre -60 dB à -80 dB.

Après avoir déterminé la puissance maximale du bruit de quantification, les largeurs de données sont optimisées en considérant une architecture ayant une granularité fine en termes de largeur de données. Les résultats sont présentés dans le tableau 3.2. Les largeurs du module de recherche de trajets, comme ceux du décodeur de symbole, dépendent du RSB. Sur une variation de 20 dB du rapport signal à bruit, la largeur de chaque opérateur varie entre 1 et 4 bits. L'économie en termes de consommation d'énergie sur une cible ASIC, illustrée dans la figure 3.22, est de 25%.

TABLE 3.2: Largeurs optimisées du module de recherche de trajets en fonction du RSB.

E_b/N_0 (dB)	1	3	5	7	9	11	13	15	17	19	21
x_{CM}	8	8	8	8	8	8	8	8	9	9	9
x_{acc}	11	11	11	11	12	12	12	13	13	13	13
x_{pow}	7	6	7	7	8	9	9	10	10	11	11

Dans la figure 3.23, la même tendance est observée pour la technique SWP. L'économie en termes de consommation d'énergie est encore plus importante : 36%. La consommation d'énergie maximale est donc supérieure de 56% par apport à la valeur minimale. Cette économie est obtenue sur un intervalle de 14 dB au lieu d'un intervalle de seulement 6 dB dans le cas de la granularité fine.

Conclusions

Dans cette section, un second module d'un récepteur WCDMA a été étudié. Ce module fournit la synchronisation au module de décodage de symbole. Il intervient indirectement

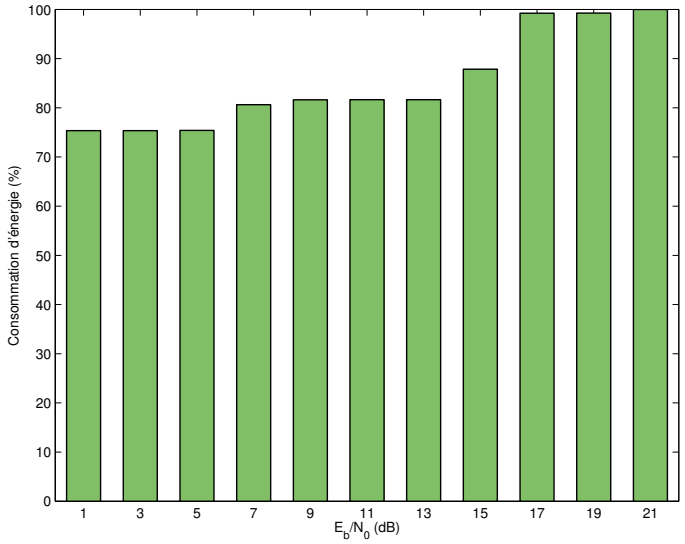


FIGURE 3.22: Consommation d'énergie pour le module de recherche de trajets.

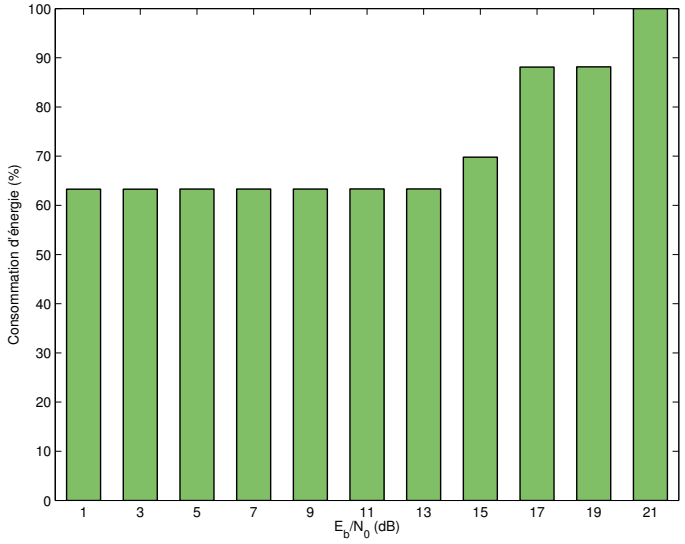


FIGURE 3.23: Consommation d'énergie pour le module de recherche de trajets avec la technique SWP.

dans le décodage des symboles et ainsi influence le taux d'erreur binaire. Cependant, il est difficile d'utiliser le taux d'erreur binaire comme métrique de performance. Les critères basés sur la théorie de la détection sont utilisés. Ils correspondent à la probabilité de non-détection et de fausse alarme. Cependant, les expérimentations montrent que les fausses alarmes sont plus sensibles au bruit de quantification que les non-détections. Ainsi, le taux de fausses alarmes est utilisé comme critère de performance.

Similaire au décodeur, la puissance maximale du bruit de quantification dépend du rapport signal sur bruit à l'entrée. Les expérimentations montrent sur un intervalle raisonnable du RSB, une différence de largeur de données de 1 à 4 bits. Au final, cette variation correspond à une évolution de 56% de la consommation d'énergie sur une cible ASIC.

3.3.4 Scénario réel

Afin de montrer l'intérêt de notre approche, il est intéressant d'estimer le pourcentage d'énergie économisée dans un scénario réel. Dans la première étape, ce pourcentage est calculé indépendamment sur chaque module, sans la connaissance de la consommation totale (filtres Nyquist, les procédures de contrôle...).

Dans le rapport technique [35] de l'ETSI sur la couche physique de l'UMTS, les critères suivants sont mentionnés :

– Effet de masquage (*shadowing*) : log-normale avec un écart-type de 10 dB ;
– Atténuation (*path-loss*) : R^{-4} dans les zones urbains et banlieue, R^{-2} à la montagne.

avec $\vec{R} = \vec{R}_0 + \vec{v}t$. En considérant la vitesse constante, $\vec{R}$ est une fonction linéaire de t.

Ces critères sont implémentés dans notre scénario de simulation de la manière suivante :

– un mobile se déplace à vitesse constante dans une zone avec $R_{\max}$ et $R_{\min}$ correspondant au RSB moyen (path-loss) de –10 dB et 5 dB ;
– un effet de masquage en log-normale avec un écart-type de 10 dB est ajouté ;
– la valeur du RSB moyen est prise pour calculer l'énergie utilisée.

Dans nos simulations, les valeurs du RSB sont dans l'intervalle [–16 dB, 10 dB] (correspondant à E_b/N_0 de –1 dB à 25 dB). La distribution du RSB suit les critères ci-dessus. Une instance du canal est illustrée à la figure 3.24. La durée de cette instance est de 60 secondes. Nous supposons qu'après 60 secondes, le rapport signal sur bruit revient à l'état initial parce que le récepteur sera connecté à une autre cellule.

Chaque valeur de RSB génère plusieurs fois un canal de Rayleigh sur un intervalle de temps de 20 ms (équivalent à 30 slots). Dans chaque réalisation du canal, le module de recherche de trajets est appelé une seule fois en supposant que le coefficient de transmission reste inchangé. Le résultat de la consommation d'énergie pour une durée de 256 chips est présenté dans le tableau 3.3 avec et sans adaptation dynamique de la précision (ADP). Le gain associé à l'ADP est fourni. Pour le module de décodage, la valeur de l'énergie totale pour 4 doigts est présentée. Pour le module de recherche des trajets, l'énergie pour l'estimation des délais d'une instance du canal est fournie.

Ce scénario permet d'obtenir une économie réelle dans le cas d'un terminal mobile en déplacement. Si le terminal est stationnaire, la consommation d'énergie dépend des conditions du canal. Si le niveau de bruit est plus important, le nombre de bits nécessaires est plus faible et l'économie sera plus faible.

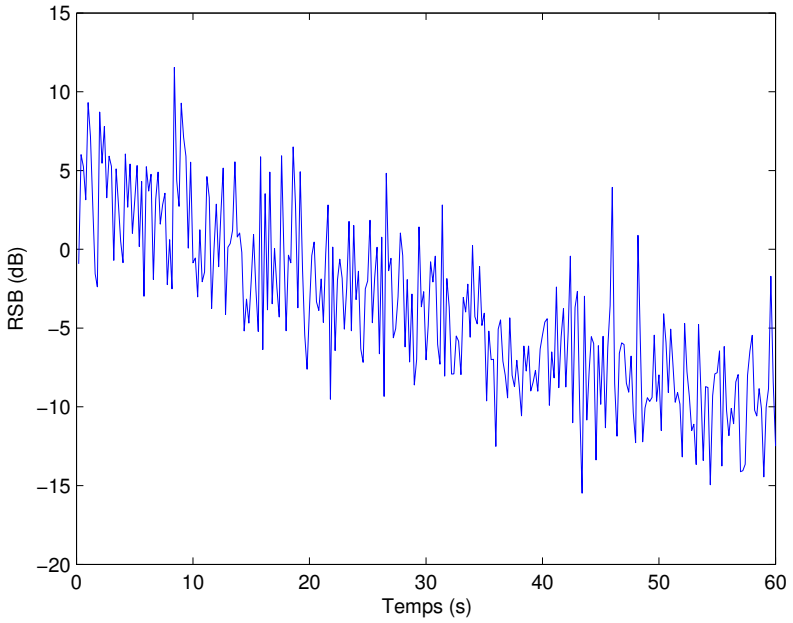


FIGURE 3.24: Variation du RSB lors du déplacement d'un utilisateur dans une cellule.

TABLE 3.3: Consommation d'énergie (Joule) du récepteur WCDMA

Module	**ADP**	**sans ADP**	**Gain**
Décodeur	$1,053 \times 10^{-7}$	$1,656 \times 10^{-7}$	36 %
Recherche de trajet	$9,883 \times 10^{-7}$	$11,45 \times 10^{-7}$	14 %

3.4 Conclusions

Dans ce chapitre, nous avons montré l'intérêt de l'adaptation dynamique des largeurs de données. Les concepts présentés dans le chapitre 2 ont été appliqués sur des systèmes de communication numérique. Les résultats montrent que le nombre total de bits nécessaires pour limiter la dégradation du TEB dépend du RSB de l'entrée du récepteur. La différence en nombre de bits entre deux valeurs extrêmes de RSB peut varier de 1 bit à 6 bits et dépend des traitements dans chaque module. De plus, le gain en terme de consommation d'énergie a été estimé dans le cas d'un scénario réel.

Deuxième partie

Optimisation des largeurs

L'optimisation correspond à la recherche de la meilleure solution d'un problème. En mathématiques, un problème d'optimisation consiste à trouver le minimum (ou maximum) d'une fonction f d'un ensemble A dans l'ensemble des nombres réels. L'ensemble A est vu comme une contrainte des solutions éligibles, dans lequel la meilleure solution est cherchée.

Dans cette partie de thèse, nous nous intéressons à une classe de problèmes d'optimisation précise : minimiser le coût sous une certaine condition de précision. L'ensemble A ne prend que des valeurs discrètes et a un nombre fini d'éléments. Cette classe de problème est appelée *optimisation combinatoire*. Généralement, le nombre des éléments de A est très grand et il est en pratique impossible de les énumérer. Le problème est donc très difficile : il n'existe pas de solution en temps polynomial.

Dans un premier temps, un état de l'art est présenté. Ceci est composé de trois étapes. Tout d'abord, nous étudions les propriétés du problème d'optimisation des largeurs. Ensuite, les algorithmes d'optimisation combinatoire sont présentés, suivis par un bilan sur les applications de ces algorithmes dans l'optimisation des largeurs. Finalement, nous donnons une perspective sur les algorithmes.

Dans un deuxième temps, nous abordons les algorithmes génétiques. Après une analyse sur les avantages et les inconvénients, nous proposons quelques améliorations. Ceux-ci sont basées sur les principes des élitismes et le couplage avec une recherche locale. Ces amélioration des algorithmes génétiques multi-objectifs permettront de trouver une solution de meilleure qualité pour un surcoût de complexité négligeable.

Dans un troisième temps, nous présentons GRASP, un algorithme combinant un déplacement intelligent avec une recherche locale. Cet algorithme n'a jamais été utilisé dans l'optimisation des largeurs. Nous proposons donc une application de GRASP permettant de trouver une solution efficacement. Les perspectives sur GRASP concluront cette partie sur l'optimisation.

Chapitre 4

État de l'art sur une classe d'algorithmes d'optimisation combinatoire

4.1 Introduction : rappel du problème à résoudre

La conversion en virgule fixe d'une application consiste à déterminer, pour chaque donnée, le nombre de bits alloués à la partie entière et à la partie fractionnaire. Ainsi, le processus de conversion en virgule fixe est composé de deux étapes principales correspondant premièrement à la détermination de la position de la virgule, et deuxièmement à l'optimisation de la largeur des données. Pour la première étape, le nombre de bits pour la partie entière d'une donnée est déterminé afin de pouvoir représenter toutes les valeurs prises par cette donnée. Pour la seconde étape, le nombre de bits pour la partie fractionnaire est déterminé. Ceci consiste à optimiser la largeur des données afin de minimiser le coût de l'implantation et d'assurer des performances minimales. Soit $\mathbf{b}$ le vecteur regroupant la largeur des entrées et de la sortie de toutes les opérations d'une application donnée. Soit $\mathcal{C}(\boldsymbol{b})$ la fonction de coût de l'application et $\Lambda(\boldsymbol{b})$ la fonction définissant les performances de l'application en fonction de la largeur des données $\mathbf{b}$. L'objectif de cette étape de détermination du nombre de bits pour la partie fractionnaire est de minimiser la fonction de coût sous contrainte que les performances restent supérieures à un seuil minimal Λ_{obj}. Ainsi, le processus d'optimisation peut être modélisé par

$$\min \mathcal{C}(\boldsymbol{b}) \quad \text{avec} \quad \Lambda(\boldsymbol{b}) \geq \Lambda_{\text{obj}}. \tag{4.1}$$

La résolution de ce problème d'optimisation nécessite la mise en œuvre de trois éléments principaux correspondant à l'évaluation de la fonction de coût, à l'évaluation de la fonction de contrainte et à l'algorithme d'optimisation. L'évaluation directe des performances d'un système en virgule fixe étant une tache trop complexe, le problème d'optimisation est modifié afin d'utiliser une métrique intermédiaire $\lambda(\boldsymbol{b})$ correspondant à la précision des calculs. Plus précisément, la métrique utilisée dans notre cas correspond au rapport signal à bruit de quantification en sortie du système. Ainsi, le problème d'optimisation présenté à l'équation (4.1) peut être transformé pour obtenir le problème modélisé par

$$\min \mathcal{C}(\boldsymbol{b}) \quad \text{avec} \quad \lambda(\boldsymbol{b}) \geq \lambda_{\min}. \tag{4.2}$$

Dans tous les chapitres de cette deuxième partie, λ et RSBQ sont mutuellement interchangeables pour représenter la métrique de précision. Dans certaines expérimentations, nous utilisons également la puissance du bruit de quantification P_b déterminée par

$$P_{b(\text{dB})} = P_{s(\text{dB})} - \text{RSBQ}_{(\text{dB})} \tag{4.3}$$

où $P_{s(\text{dB})}$ et $P_{b(\text{dB})}$ sont respectivement la puissance du signal et la puissance du bruit de quantification exprimées en dB.

4.1.1 Contexte du problème d'optimisation

Influence de l'architecture sur le domaine de définition des variables du problème d'optimisation

Les architectures ciblées par notre approche correspondent principalement à deux classes : les architectures dédiées (ASIC, FPGA, éventuellement ASIP) permettent une grande flexibilité sur les tailles des données et les architectures de processeur (microprocesseur, DSP, extensions SIMD) possèdent un nombre limité de possibilités sur la largeur des données.

Dans le cas d'une architecture existante, la largeur des données est fixée par la taille des opérateurs disponibles au sein de celle-ci. Le domaine de définition dépend des opérations supportées par l'architecture. Les opérations supportées sont classées en trois types correspondant aux opérations classiques, multi-précision ou utilisant le parallélisme au niveau données (SIMD/SWP).

- Pour une opération classique, la largeur des opérandes de l'opération est égale à la largeur des entrées et de la sortie d'un opérateur. Ainsi, un seul opérateur est utilisé pour exécuter l'opération, et la latence de l'opération correspond à celle de l'opérateur.
- Les opérations multi-précisions permettent de traiter des données dont la largeur est supérieure à la largeur des entrées et de la sortie d'un opérateur. Dans ce cas l'opération est décomposée en une suite d'opérations classiques. Cette association d'opérations est soit temporelle soit spatiale. Dans le premier cas, un seul opérateur est utilisé pour exécuter la suite d'opérations classiques les unes après les autres. Ainsi, la latence de l'opération est supérieure à celle de l'opérateur et donc à celle d'une opération classique. Dans le second cas, plusieurs opérateurs sont utilisés en parallèle pour exécuter les différentes opérations. En conséquence, le nombre de ressources utilisées est supérieur à celui obtenu dans le cas d'une opération classique.
- Les opérations SWP (ou SIMD) utilisent des données dont la largeur est inférieure à la largeur des entrées et de la sortie d'un opérateur en vue d'exécuter en même temps plusieurs opérations en parallèle sur le même opérateur. La latence de l'opération correspond à celle de l'opérateur donc d'une opération classique. Mais plusieurs opérations sont réalisées en parallèle et ainsi la durée globale du traitement est diminuée. Un compromis *précision–performance* doit donc être recherché.

Dans le cas de la conception d'une architecture dédiée à l'application ou au domaine de l'application, la largeur des opérateurs est à définir par le concepteur. Ainsi, le domaine de définition des variables d'optimisation correspond potentiellement à l'ensemble $\mathbb{N}^*$. Cependant, celui-ci est restreint à des valeurs raisonnables, par exemple de 4 à 40 bits.

Type de solution recherchée

L'objectif du problème d'optimisation des largeurs est de trouver une solution au problème d'optimisation présenté à l'équation 4.2. La conversion en virgule fixe d'une application donnée va se traduire par la détermination d'une contrainte de précision puis par l'optimisation de la largeur des données pour cette contrainte. Dans notre cas, nous souhaitons adapter dynamiquement la spécification virgule fixe en fonction de conditions externes. Ainsi, différentes contraintes de précision doivent être prises en compte. En conséquence, dans ce cas l'objectif est de trouver la solution du problème d'optimisation pour plusieurs contraintes de précision. Si le nombre de contraintes de précision est relativement élevé, ceci conduit à trouver la *frontière de Pareto* de la relation entre le coût de l'implantation et la précision des calculs. Cette frontière de Pareto est aussi utilisée dans le cadre de la conception au niveau système [117]. Dans ce contexte, le système complet est composé de différents blocs et l'optimisation globale au niveau système nécessite la connaissance de la frontière de Pareto du coût en fonction de la précision de chaque bloc composant le système.

En conclusion, l'objectif est soit de trouver la solution du problème d'optimisation pour un faible nombre de contraintes de précision ou soit d'obtenir la frontière de Pareto du coût en fonction de la précision des calculs. Ainsi, les méthodes permettant d'obtenir directement la frontière de Pareto mais nécessitant des temps d'exécution élevés, trouveront leur place pour le second cas de figure présenté ci-dessous. En effet, ces méthodes peuvent être plus rapides que l'utilisation de méthodes classiques nécessitant de réaliser une optimisation pour chaque point de la courbe.

Critères de sélection des algorithmes d'optimisation

L'analyse et la comparaison des méthodes d'optimisation se basent sur deux critères : la qualité de la solution obtenue et la rapidité d'obtention de cette solution. L'objectif principal de ce problème d'optimisation est d'obtenir la solution de meilleure qualité en un temps raisonnable. Le concepteur peut accepter que le temps d'exécution de ce processus d'optimisation soit plus important s'il peut réduire le coût de son implantation. Cependant, des algorithmes d'optimisation, dont la qualité de la solution est dégradée mais dont le temps d'exécution est faible, peuvent trouver leur place dans le cas où le concepteur souhaite obtenir rapidement une première approximation du coût de son implantation pour une contrainte de précision donnée.

Ainsi, l'objectif de ce travail est aussi de mesurer ces deux critères de comparaison (qualité de la solution et temps d'exécution) pour les différents algorithmes étudiés. Ensuite, le concepteur pourra choisir en fonction de son critère de choix l'algorithme le plus adéquat pour résoudre son problème.

4.1.2 Classification des algorithmes

Solutions optimales et sous-optimales

Les algorithmes de recherche peuvent être classés en deux catégories : les *méthodes exactes* (complètes) garantissant la complétude de la résolution et les *méthodes approchées*

(incomplètes) permettant d'obtenir une solution sous-optimale avec une certaine efficacité.

Les méthodes exactes énumèrent, souvent de manière implicite, toutes les solutions et utilisent des techniques pour détecter le plus tôt possible, lors de l'exploration des solutions, les échecs. Ces méthodes sont développées depuis une quarantaine d'année et sont basées sur les algorithmes de séparation et évaluation progressive (SEP, ou encore BaB pour *Branch and Bound*) ou les algorithmes avec retour arrière. Les méthodes exactes sont capables de trouver les solutions optimales.

Des implémentations parallèles de SEP ont été proposées récemment [47, 29]. Elles permettent d'accélérer la recherche. Malheureusement, le problème d'optimisation des largeurs est NP–difficile [28]. Par conséquent, le temps de calcul risque d'augmenter exponentiellement avec la taille du problème, et les méthodes exactes deviennent très vite inefficaces pour des applications de taille importante (donc correspondant à la réalité).

Les méthodes approchées ne garantissent pas l'optimalité mais permettent de résoudre des problèmes de grande taille en un temps raisonnable. Depuis les années 1990, des méthodes générales (*métaheuristiques*) [63] sont développées. Une métaheuristique permet de construire des méthodes heuristiques pour un problème ou une classe de problèmes d'optimisation. Ces méthodes ont montré leur efficacité sur un nombre important de problèmes d'optimisation classiques. Il est possible d'avoir un compromis entre la qualité des solutions et le temps de calcul en réglant les paramètres d'une métaheuristique. Cependant, ces méthodes ne garantissent pas l'optimalité de la solution trouvée, ni de trouver une solution optimale.

Les objectifs des méthodes exactes et des méthodes approchées ne sont pas identiques. Dans certaines conditions, ces méthodes peuvent coopérer. La performance de l'algorithme dépend du problème d'optimisation. Ainsi, dans [14], une comparaison de 9 méthodes d'optimisation de la largeur des données sur 12 applications DSP de taille petite et moyenne a été effectuée. Le résultat montre que même pour ce type de problème bien défini, aucune méthode n'est meilleure que les autres sur l'ensemble des applications. De plus, dans [150], les auteurs montrent la difficulté à tirer des conclusions de la comparaison des algorithmes d'optimisation sur différents problèmes. Un algorithme plus performant sur une classe de problème peut être de moins bonne qualité sur une autre classe. Cela nécessite la connaissance des algorithmes et des problèmes pour pouvoir retenir une bonne méthode d'optimisation.

Algorithmes déterministes et algorithmes aléatoires

Les algorithmes d'optimisation peuvent être classés en deux sous-ensembles : les algorithmes déterministes et les algorithmes aléatoires. Les algorithmes déterministes utilisent la connaissance de la fonction de coût et de la fonction de performance pour progresser dans une direction de façon déterministe. Les critères choisis pour la progression peuvent être la variation de fonctions, une combinaison de variations, ou encore le gradient.

Les algorithmes aléatoires n'utilisent pas directement ces critères. A chaque itération de l'algorithme d'optimisation, les solutions sont générées aléatoirement et les meilleures sont conservées pour la prochaine itération.

4.1.3 Définitions

Dans la suite du document, les notations et les définitions suivantes sont utilisées.

Notation 4.1. *Dans l'application à optimiser, N est le nombre de variables du problème d'optimisation, b_i avec $1 \leq i \leq N$ est la largeur de l'opérande i et $\boldsymbol{b} = [b_1, b_2, ..., b_N]$ est le vecteur regroupant l'ensemble (la combinaison) des largeurs.*

Notation 4.2. *La distance entre deux vecteurs des largeurs $\boldsymbol{b^j}$ et $\boldsymbol{b^k}$ est définie par*

$$d(\boldsymbol{b^j}, \boldsymbol{b^k}) = \sum_{i=1}^{N} |b_i^j - b_i^k|. \tag{4.4}$$

Définition 4.1. *La largeur minimale absolue d'une opération (d'un opérande)* op_k *est définie comme la largeur minimale de cette opération (cet opérande), telle que la contrainte de précision sur l'application soit toujours satisfaite, lorsque toutes les autres largeurs sont à une valeur maximale prédéfinie. La* ***combinaison des largeurs minimales*** *(CLM) correspond au vecteur $\boldsymbol{b}^{\min} = (b_1^{\min}, b_2^{\min}, ..., b_N^{\min})$ où chaque élément $b_k^{\min}$ correspond à la largeur minimale de l'opération* op_k.

La combinaison des largeurs minimales est souvent la valeur initiale des algorithmes de recherche. Dans plusieurs cas, le résultat trouvé est distant uniquement d'un ou de deux bits de cette valeur.

Définition 4.2. *Une opération (un opérande) ne peut prendre une largeur plus grande que $b_k^{\max}$ dépendant de l'architecture. Dans la plupart des cas, cette valeur $b_k^{\max}$ est fixée à 32 bits. La* ***combinaison des largeurs maximales*** *est définie par $\boldsymbol{b}^{\max} = (b_1^{\max}, b_2^{\max}, ..., b_N^{\max})$*

Définition 4.3. *La* ***largeur uniforme minimale*** *est la valeur minimale b^{uni} tel que $\boldsymbol{b}^{\mathrm{uni}} = (b^{\mathrm{uni}}, b^{\mathrm{uni}}, ..., b^{\mathrm{uni}})$ satisfasse la contrainte de précision.*

Pour faciliter la présentation, nous définissions les fonctions **succ** et **pred**. Ces fonctions déplacent la valeur actuelle $\boldsymbol{b}$ de respectivement $+1$ et -1 bit à la position k selon

$$\boldsymbol{b}^{\mathrm{succ}} = \mathrm{succ}(\boldsymbol{b}, k) \Leftrightarrow \begin{cases} b_j^{\mathrm{succ}} = b_j & \text{si} \quad j \neq k \\ b_k^{\mathrm{succ}} = b_k + 1 \end{cases} \tag{4.5}$$

$$\boldsymbol{b}^{\mathrm{pred}} = \mathrm{pred}(\boldsymbol{b}, k) \Leftrightarrow \begin{cases} b_j^{\mathrm{pred}} = b_j & \text{si} \quad j \neq k \\ b_k^{\mathrm{pred}} = b_k - 1 \end{cases} \tag{4.6}$$

4.2 Algorithmes d'optimisation combinatoire

Considérons le problème d'optimisation présenté à l'équation 4.2. Dans ce contexte, $\boldsymbol{b}$ est un vecteur discret, $\lambda(\boldsymbol{b})$ et $\mathcal{C}(\boldsymbol{b})$ sont des fonctions pouvant être monotones non-décroissantes sous certaines conditions. La recherche des largeurs de données optimales est un problème d'optimisation discrète, sous contrainte et multi-variables. Cette classe de problèmes est appelée *optimisation combinatoire*. Bien que les problèmes d'optimisation combinatoire soient faciles à définir ainsi qu'à résoudre en énumérant l'ensemble des solutions, le temps nécessaire pour résoudre le problème est souvent prohibitif et des algorithmes d'optimisation en temps raisonnable sont nécessaires.

4.2.1 Algorithmes déterministes

Recherche locale

Les algorithmes de recherche locale permettent de trouver un minimum local à partir d'un point de départ. L'algorithme part d'une solution candidate et se déplace de façon itérative vers une solution voisine. Il est donc nécessaire de définir la notion de "voisinage" dans l'espace de recherche. Pour un problème d'optimisation des largeurs, ce concept peut être défini de la manière suivante : deux configurations sont voisines d'une distance de k si leurs largeurs sont différentes d'au plus k bits au total. La solution optimale avec un voisinage k est appelée k-optimale.

Une recherche locale peut s'arrêter après un nombre fixé d'itérations ou lorsque la meilleure solution trouvée par l'algorithme n'a pas été améliorée depuis un nombre donné d'itérations. Même dans ce cas d'impossibilité d'améliorer la solution courante, il est probable que la solution trouvée ne soit pas la meilleure dans l'espace de recherche. Les algorithmes de recherche locale sont alors des algorithmes sous-optimaux.

Recherche avec tabous (*Tabu Search*)

La recherche avec tabous [50, 51] est une métaheuristique d'optimisation devant être utilisée en coopération avec d'autres méthodes. Dans plusieurs cas, le couplage de l'algorithme glouton avec la recherche avec tabous permet d'avoir une meilleure recherche locale et permet d'éviter des déplacements inutiles. Le détail de cet algorithme est présenté en section 6.1.

Séparation et évaluation

L'algorithme *séparation et évaluation* (SEP ou aussi *BaB* pour *Branch and Bound*) est une méthode d'énumération implicite. La connaissance des propriétés du problème permet d'énumérer seulement les solutions potentiellement bonnes. Cet algorithme est une méthode générale. Cet algorithme peut s'appliquer individuellement ou en combinaison avec d'autres méthodes. Par exemple, un problème d'optimisation combinatoire peut être résolu dans $\mathbb{R}^N$, puis la séparation et évaluation est utilisée pour trouver la solution correspondant avec une complexité $\mathcal{O}(2^N)$.

Programmation linéaire en nombres entiers

Afin de mettre le problème d'optimisation dans le contexte d'une programmation linéaire en nombres entiers (PLNE ou ILP pour *Integer Linear Programming*), ou d'une programmation linéaire mixte (PLM ou MILP pour *Mixed Integer Linear Programming*), la fonction de coût et la fonction de contrainte doivent être approchées sous une forme linéaire. Ensuite, comme la PLM (ainsi que PLNE) est un problème NP–difficile, il est nécessaire d'utiliser une heuristique pour résoudre ce problème. L'algorithme le plus largement utilisé est la séparation et évaluation.

4.2.2 Algorithmes aléatoires

Recuit simulé

Le recuit simulé (SA pour *Simulated Annealing*)[83] est une métaheuristique inspirée de l'évolution d'un système thermodynamique en métallurgie. Chaque point de l'espace de recherche correspond à un état du système. Le but est d'amener le système à un état avec l'énergie interne la plus faible, en alternant des cycles de chauffage, maintien en température puis refroidissement lent. Comme cet algorithme est largement utilisé dans l'optimisation de largeurs, celui-ci est présenté de manière générale à travers l'algorithme 4.1. s_0 est l'état initial ou la solution initiale, s est l'état actuel, s_n est un des états suivants et s_b est le meilleur état trouvé. e_0, e, e_n et e_b sont respectivement l'énergie de chaque état. k est le nombre d'évaluations d'énergie effectuées et k_max le nombre maximal d'évaluations autorisé. La fonction $E(s)$ permet d'évaluer l'énergie associée à l'état s.

Cet algorithme s'appuie d'une part sur une fonction d'évaluation du recuit $P()$ (généralement exponentielle) qui est fonction de l'évolution du coût et de l'évolution de la température $temp()$. La condition $P(e, e_n, \mathrm{temp}(k/k_\mathrm{max})) > \mathrm{random}()$ permet non seulement de se déplacer vers un meilleur état mais aussi de prendre du recul avec une certaine probabilité.

Algorithme 4.1 Recuit simulé.

```
s ← s_0                                          état initial
e ← E(s)                                         énergie associée à l'état initial
s_b ← s                                          meilleur état trouvé
e_b ← e                                          meilleure énergie obtenue
k ← 0                                            nombre d'évaluations d'énergie effectuées
tantque k < k_max et e > e_max faire             k_max : nombre maximal d'évaluation permis
  s_n ← voisin(s)                                fonction de déplacement
  e_n ← E(s_n)
  si e_n < e_b alors
    s_b ← s_n                                    nouvelle solution
    e_b ← e_n
  fin si
  si P(e, e_n, temp(k/k_max)) > random() alors   sélection ?
    s ← s_n                                      se déplace vers cet état
    e ← e_n
  fin si
  k ← k + 1
fin tantque
```

Dans de nombreux cas, le but de cet algorithme est seulement de trouver une solution acceptable dans un temps fixe plutôt que de trouver la meilleure solution. Les exemples de la mise en œuvre de cet algorithme sont dans [10, 17, 43, 77, 124, 66].

L'inconvénient principal du recuit simulé réside dans le choix des paramètres (l'état initial, le programme de recuit P, la loi de décroissance de la température, etc.) et le temps de convergence. Une version améliorée d'Ingber, le recuit simulé adapté (*Adaptive Simulated Annealing* – ASA ou également *Very Fast Simulated Annealing*), donne de meilleures performances [76]. Cette méthode consiste à ne pas utiliser la distribution de Boltzmann

et à adapter la fonction de sélection pendant le processus d'optimisation. Des expérimentations montrent que l'algorithme ASA converge plus rapidement que la version d'origine de SA.

Algorithmes génétiques

Les algorithmes génétiques [52] sont des métaheuristiques, permettant d'obtenir une solution approchée en un temps raisonnable ou fixé. Ces algorithmes utilisent la notion de sélection naturelle définie par Darwin. D'abord, la population est créée à partir des solutions potentielles au problème. Ensuite, le traitement évolutionnaire modifie la population et génère les générations pour l'itération suivante. Finalement, les individus dominants sont sélectionnés afin d'obtenir une meilleur solution.

Procédure de recherche gloutonne aléatoire et adaptative (GRASP)

Les algorithmes GRASP [126] *(Greedy Randomized Adaptive Search Procedures)* sont des métaheuristiques itératives. Ils sont composés d'un algorithme glouton aléatoire et d'une recherche locale. GRASP est donc une méthode hybride. Dans la première étape, une solution est itérativement construite en ajoutant les éléments dans chaque itération. Ces éléments sont choisis aléatoirement à partir d'une liste obtenue par l'algorithme glouton. Dans la seconde étape, une recherche locale est utilisée pour améliorer la solution. Ces deux étapes sont répétées itérativement, et, à la fin, l'algorithme retourne la meilleure solution trouvée.

GRASP est utilisé dans plusieurs domaines et récemment dans le domaine de traitement du signal. Skaf et al. [137] utilisent GRASP pour déterminer directement les coefficients en représentation chiffres signés canoniques (CSD) d'un filtre. Dans [85], Kountouris réalise la synthèse de fréquence numérique avec une méthode répétitive aléatoire. Cependant, GRASP n'a jamais été appliqué dans l'optimisation des largeurs de données. Nous présenterons notre adaptation de GRASP dans le chapitre 6.

4.3 Optimisation des largeurs

Dans cette section, l'application des algorithmes d'optimisation combinatoire est étudiée pour l'optimisation des largeurs des données d'un système. Tout d'abord, les algorithmes déterministes sont détaillés. Ensuite, les algorithmes aléatoires sont présentés.

4.3.1 Algorithmes déterministes

Recherche exhaustive

La méthode proposée dans [144] et décrite dans l'algorithme 4.2 réalise une recherche exhaustive dans un sous-espace de l'espace de recherche. L'algorithme débute par la CLM ($\boldsymbol{b}^{\min}$), ensuite un bit est temporairement ajouté à un opérateur et la contrainte de précision est vérifiée. Si cette contrainte n'est pas satisfaite, alors le nombre d de bits total ajoutés à la CLM et distribués aux différentes opérations est incrémenté, et la procédure est répétée tant que la contrainte de précision n'est pas satisfaite.

Algorithme 4.2 Recherche exhaustive.

1: $d \leftarrow 1$ *distance entre $\boldsymbol{b}^{\min}$ et la solution actuellement recherchée*
2: **tantque vrai faire**
3: **pour tout** $\boldsymbol{c} = [c_1, c_2, ..., c_N]$ tel que $\sum c_i = d$, $c_i \geq 0$ **faire**
4: **si** $\mathrm{RSBQ}(\boldsymbol{b}^{\min} + \boldsymbol{c}) \geq \mathrm{RSBQ}_{\min}$ **alors**
5: $\boldsymbol{b} \leftarrow \boldsymbol{b}^{\min} + \boldsymbol{c}$
6: **return** $\boldsymbol{b}$
7: **fin si**
8: **fin pour**
9: $d \leftarrow d + 1$
10: **fin tantque**

Malgré le terme « exhaustif », l'algorithme n'explore pas entièrement l'espace de recherche (ligne 4–7). Il garantit de trouver une solution, mais la solution trouvée est sous-optimale. Le nombre d'évaluations de la précision est

$$\begin{aligned} \Gamma_{N+1}^{d-1} &= \mathrm{C}_{N+d-1}^{d-1} \\ &= \frac{(d+N-1)(d+N-2)...(d+2)(d+1)d}{N!} \end{aligned} \tag{4.7}$$

où d est la distance entre la solution et la CLM. La valeur maximale de d est obtenue à partir de $d + b_k^{\min} \leq b_k^{\max}$.

Cette méthode ne prend pas en compte le coût réel (ou estimé) de l'implantation pour décider des opérations sélectionnées pour augmenter la largeur. Ainsi, certaines opérations très coûteuses peuvent être augmentées au détriment d'opérations moins coûteuses.

Algorithme glouton

Le principe de l'algorithme glouton *(greedy search)* est de trouver un optimum local dans chaque étape, jusqu'à la détermination de la solution. Il existe plusieurs variantes de

cet algorithme dont une consiste à maximiser la précision à chaque étape. L'Algorithme 4.3 décrit son fonctionnement.

Algorithme 4.3 Glouton.

$\boldsymbol{b} \leftarrow \boldsymbol{b}^{\min}$
tantque $\mathrm{RSBQ}(\boldsymbol{b}) < \mathrm{RSBQ}_{\min}$ **faire**
  pour $k = 1$ to N **faire**
    $q_k \leftarrow \mathrm{RSBQ}(\mathrm{succ}(\boldsymbol{b}, k))$
  fin pour
  $i \leftarrow \arg\max q_k$ *ou bien,* $\arg\max q_k - \mathrm{RSBQ}(\boldsymbol{b})$
  $b_i \leftarrow b_i + 1$
fin tantque

Cet algorithme est aussi appelé *min + 1 bit* ou *recherche séquentielle* par K. Han [57]. Pour éviter les maxima locaux et converger plus rapidement, la méthode *min + b bits*, où b bits sont distribués en total dans chaque étape, est considérée. D'autres variantes de *min + 1 bit* sont les algorithmes *max – 1 bit* et *max – b bits* [115], commençant par la combinaison des largeurs maximales $\boldsymbol{b}^{\max}$. Les algorithmes *min + 1 bit* et *max – 1 bit* garantissent que les solutions sont 1–optimales, [1] mais les variantes *min + b* et *max - b* ne garantissent rien.

L'algorithme glouton peut être utilisé seul ou en coopération avec d'autres algorithmes [95]. Dans la plupart des cas, une solution sous-optimale peut être trouvée rapidement. Si une solution à une distance d est trouvée, le nombre d'évaluations est $N \times d$. Dans le pire cas, le nombre d'étapes est égal à

$$\sum_{k=1}^{N} (b_k^{\max} - b_k^{\min}). \tag{4.8}$$

Dans [61], Han et Evans proposent d'utiliser la *sensibilité*. En ayant un bon compromis entre la fonction de coût (« complexité ») et la sensibilité (RSBQ dans le cas général, ou directement les performances du systèmes en termes de qualité).

Dans [57], une variante de *min + 1 bit*, la *pré-planification*, est présentée. Au lieu de calculer N gradients à chaque étape, la valeur du gradient calculé pour la CLM est utilisée : le gradient de b_k est calculé avec les autres opérateurs fixés à leurs valeurs maximales. Dans ce cas, chaque opération est traitée indépendamment. Ainsi, le gradient $\nabla_k(\boldsymbol{b})$ d'une opération k est donc considéré indépendant des autres opérations, ce qui donne

$$\begin{aligned} \nabla_k(\boldsymbol{b}) &= \nabla_k(b_1, ..., b_{k-1}, b_k, k_{k+1}, ..., b_N) \\ &\approx \nabla_k(b_1^{\max}, ..., b_{k-1}^{\max}, b_k, b_{k+1}^{\max}, ..., b_N^{\max}) = f_k(b_k). \end{aligned} \tag{4.9}$$

Cette méthode est plus rapide mais elle converge rarement vers une aussi bonne solution que la version originale.

Pour améliorer la recherche de la meilleure direction à suivre à chaque étape de l'algorithme glouton, il est possible de prendre en compte la fonction de coût en utilisant

1. Cette solution est optimale par rapport à toutes les autres valeurs ayant une distance maximale de k égal à 1 bit.

une métrique correspondant au rapport entre le gain sur la précision et l'augmentation du coût.

L'algorithme 4.4 proposé dans [70] introduit un paramètre K limitant le nombre maximal de solutions explorées à chaque étape. L'algorithme mémorise toutes les solutions permettant de franchir la contrainte de précision. La procédure termine lorsque l'ensemble des solutions atteint la limite K. Cette modification garantit que tous les candidats testés lors des différentes itérations, non seulement ceux qui sont sur le chemin, sont pris en compte. Le résultat est donc meilleur, ou au pire égal, à celui obtenu avec un algorithme glouton classique.

Algorithme 4.4 Algorithme proposé dans [70]

$\boldsymbol{b} \leftarrow \boldsymbol{b}^{\min}$
$q' \leftarrow \mathrm{RSBQ}(\boldsymbol{b})$; $c' \leftarrow \mathcal{C}(\boldsymbol{b})$
$S \leftarrow \emptyset$ — *S : ensemble de solutions*
$K \leftarrow 5$ — *K : nombre maximal de solution*
tantque $|S| < K$ **faire** — *$|S|$: nombre d'éléments (solutions) dans S*
  pour $k = 1$ to N **faire**
    $q_k \leftarrow \mathrm{RSBQ}(\mathrm{succ}(\boldsymbol{b}, k))$
    $c_k \leftarrow \mathcal{C}(\mathrm{succ}(\boldsymbol{b}, k))$
    si $q_k > \mathrm{RSBQ}_{\min}$ **alors**
      $S \leftarrow S + \mathrm{succ}(\boldsymbol{b}, k)$
    fin si
  fin pour
  $\nabla = (\vec{q} - q')/(\vec{c} - c')$ — *gradient, pas fixé à 1 bit*
  $i \leftarrow \arg\max \nabla_k$
  $b_i \leftarrow b_i + 1$
  $(q', c') \leftarrow (q_i, c_i)$
fin tantque
$\boldsymbol{b}^{\mathrm{opt}} \leftarrow \arg\min_{\mathbf{b} \in S} \mathcal{C}(\boldsymbol{b})$

Procédure heuristique

Cette heuristique a été proposée par W. Sung dans [144]. Le but est de limiter fortement le nombre d'itérations afin de réduire le nombre d'évaluations du coût et de la précision. Cette approche est justifiée par le fait que l'évaluation de la précision est réalisée par simulation. A partir de la CLM, toutes les largeurs sont augmentées en même temps jusqu'à ce que la contrainte de précision soit satisfaite. Cette étape ne prend normalement que quelques simulations. L'étape suivante va réduire une largeur, en commençant par l'opération la plus coûteuse. Cette étape est répétée tant que la contrainte est toujours satisfaite. Cette étape, équivalente à un *max – 1 bit*, nécessite N évaluations de la précision. Cette procédure suppose que la solution est proche de la CLM mais cette hypothèse n'est pas toujours valide.

Séparation et évaluation

Dans [20], l'algorithme de séparation et évaluation est utilisé directement, mais ceci n'est valable que pour des problèmes de taille petite et moyenne, en raison de sa complexité exponentielle. Plusieurs techniques d'amélioration sont possibles pour limiter l'espace de recherche. L'algorithme proposé par Choi réalise les étapes suivantes :

- recherche de la largeur uniforme minimale b^{uni} ;
- recherche de la combinaison des largeurs minimales $b_k^{\min}$;
- application de l'algorithme SEP dans les intervalles de b^{uni} à $b_k^{\min}$. Si pour un nœud $\boldsymbol{b}^t$ la contrainte de précision n'est pas satisfaite, les nœuds $\boldsymbol{b}^k$ suivants ($b_n^k \leq b_n^t \quad \forall n = 1..N$) sont ignorés.

Dans [96], quelques techniques sont proposées pour limiter l'espace de recherche. Ces travaux se situent dans le contexte de l'optimisation des largeurs dans le cas d'architectures ayant une granularité moyenne en termes de largeurs supportées. Premièrement, l'influence de chaque opération est mesurée. Cette information peut-être déterminée par le nombre d'exécutions de l'opération. Deuxièmement, avant d'explorer une branche, le coût minimal et la précision maximale de cette branche sont calculés. Si le coût minimal est supérieur à celui de la solution actuelle, ou si la précision maximale n'est pas satisfaite, cette branche est ignorée. Troisièmement, la valeur approximée réelle de la solution est trouvée grâce à une programmation par contraintes, puis les variables ne prennent que des valeurs autour de la solution.

Programmation linéaire en nombres entiers

Constantinides propose cet algorithme dans [25] et le détaille dans [26]. Sachant que le temps d'optimisation peut être très long pour des applications ayant un nombre de variables élevé, cet algorithme est plutôt utilisé pour comparer différents algorithmes d'optimisation sur des applications peu complexes.

4.3.2 Algorithmes aléatoires

Recuit simulé

L'algorithme général de cette méthode est présenté dans la partie 4.2.2. Le recuit simulé est largement utilisé dans le domaine du traitement du signal [18] et ainsi que dans l'optimisation de largeur de données. Dans [17] les auteurs transforment le problème d'optimisation des largeurs en une programmation linéaire en nombres entiers et utilisent le recuit simulé pour résoudre ce problème. Dans [19], pour optimiser un régulateur PID, Chen et al. résolvent un problème d'optimisation non-linéaire en valeurs réelles à l'aide d'un algorithme de recuit simulé adapté. Ensuite, le nombre de bits est approximé à partir de la valeur réelle.

Dans [90], Lee et al. proposent d'utiliser le recuit simulé adapté, avec la largeur uniforme minimale comme valeur initiale. Des approximations polynomiales avec des degrés de 2 à 12 sont utilisées pour évaluer la performance de différents algorithmes d'optimisation. La différence entre le résultat sous-optimal d'un recuit simulé (temps d'optimisation : quelques secondes) et le résultat optimal d'une programmation linéaire en nombres entiers

(temps : quelques heures à quelques jours) est de moins de 1 %. Il faut noter que la différence entre ces solutions et la largeur uniforme minimale est de 10 %. Les auteurs aussi remarquent que pour les *designs* les plus complexes, l'espace de recherche sera vaste et le temps d'optimisation sera long, voire prohibitif.

Algorithmes génétiques

Les algorithmes génétiques sont basés sur une population et n'intègrent pas de contrainte. La contrainte de précision est donc transformée en objectif. Le problème d'optimisation est alors à objectifs multiples : à la fois minimiser le coût et minimiser la puissance de bruit de quantification. Un état de l'art détaillé est présenté dans la partie 5.1.4 du chapitre dédié aux algorithmes génétiques.

4.3.3 Bilan

Jusqu'à maintenant, une vue d'ensemble des méthodes d'optimisation a été présentée. Il est intéressant de dresser un bilan en triant les méthodes par les laboratoires où elles ont été définies. Cette liste est non-exhaustive.

Dans [156] (Chalmers University of Technology, Sweden), un SIC *(Successive Interference Canceller)* pour WCDMA est étudié. L'implémentation virgule fixe est modélisée par une source de bruit à l'entrée. L'idée est de considérer cette implémentation comme une implémentation en virgule flottante avec un RSB modifié. Pourtant, le modèle est très simple. Le système utilise deux pas de quantification Δ et Δ_1 avec $\Delta_1 = \Delta/8$ dans 6 sources de quantification. La source de bruit en entrée est une fonction de L (nombre de trajets, constant du point de vue virgule fixe), de Δ et de Δ_1, ce qui revient donc à une fonction du seul scalaire Δ. La précision est évaluée directement à partir de Δ et aucune optimisation n'est requise.

K.I. Kum et W. Sung (Seoul National University, Taiwan) travaillent dans ce domaine depuis 1994. Dans un de leurs premiers articles, une recherche exhaustive et une *procédure heuristique* sont proposées [144]. Dans [87, 88], Kum a proposé de regrouper les opérateurs avant d'optimiser les largeurs. Un filtre elliptique du cinquième ordre avec 36 opérateurs est regroupé en 17 *clusters* par une heuristique de partition de cliques. Ensuite, la même méthode d'optimisation est appliquée. 44 simulations sont requises pour 17 largeurs. Ce travail est étendu et représenté dans [88]. L'ordonnancement et l'optimisation des largeurs utilisent la programmation linéaire en nombres entiers, ce qui permet de trouver la solution optimale.

Constantinides (Imperial College London, UK) et al. travaillent sur la détermination des largeurs depuis 2001 [133, 90, 27, 25, 22, 24, 28, 23, 13], mais la procédure d'optimisation est décrite dans les premiers articles. Dans [23], un algorithme glouton *max – 1* est proposé. L'auteur suppose que la fonction de coût et la contrainte de précision ne sont pas monotones par rapport à n'importe quel paramètre b_j. Cependant, les fonctions sont monotones pour la largeur uniforme. Cette hypothèse permet de déterminer la largeur uniforme minimale $\boldsymbol{b}^{\text{uni}}$ par une recherche binaire. Avec cette hypothèse, l'auteur ne peut pas garantir que la solution soit inférieure à $\boldsymbol{b}^{\text{uni}}$, l'optimisation commence donc à partir de $k\boldsymbol{b}^{\text{uni}}$ avec k facteur d'échelle ($k = 2$ dans leurs expérimentations). En plus, dans chaque

étape, au lieu de calculer le coût de la prochaine valeur $b_k = b_k^{\text{old}} - 1$ par

$$q_k = \lambda(\boldsymbol{b} \mid b_k = b_k^{\text{old}} - 1) \tag{4.10}$$

il calcule le meilleur coût selon

$$q_k = \min_{\lambda(\boldsymbol{b}) \geq \lambda_{\min}} \lambda(\boldsymbol{b} \mid b_k = 1..b_k^{\text{old}}). \tag{4.11}$$

Les résultats de cet algorithme sont comparés avec les solutions optimales trouvées par la programmation linéaire en nombres entiers, dans les cas de petits systèmes, la différence n'est en moyenne que de 0,7 %[24]. Un modèle pour la programmation linéaire en nombres entiers est aussi présenté dans cet article.

Dans [61], Han et Evans (University of Texas, Austin, USA) proposent une amélioration d'un algorithme glouton qui donne de meilleurs résultats. Ces auteurs proposent également un algorithme génétique en utilisant une boîte à outils GEATbx sous MATLAB. Ces méthodes nécessitent une fonction à objectifs multiples (coût, RSBQ), et, pour l'algorithme génétique, le nombre maximal de générations.

Finalement, dans notre équipe à l'IRISA, Ménard et al. (Université de Rennes) ont proposé des algorithmes d'optimisation efficaces par séparation et évaluation [96] et descente de gradient [70].

Auteurs	Établissement	Méthodes
Kum, Sung	Séoul	recherche exhaustive, procédure heuristique
Cantin et al.	Montréal	algorithmes glouton, procédure heuristique, recherche exhaustive, recuit simulé, pré-planification, évaluation et séparation
Constantinides et al.	Londres	algorithme glouton, programmation linéaire en nombres entiers
Han, Evans	Austin	pré-planification, descente de gradient modifiée, algorithme génétique
Lee et al.	Los Angeles	recuit simulé
Ménard et al.	Lannion	séparation et évaluation, descente de gradient

TABLE 4.1: Tableau récapitulatif des principales méthodes d'optimisation utilisées dans la littérature

4.4 Conclusions

Dans ce chapitre, nous avons présenté le problème d'optimisation des largeurs ainsi qu'un état de l'art de ces algorithmes. Plusieurs algorithmes d'optimisation combinatoire ont été étudiés. Les performances de chaque algorithme dépendent du problème et peuvent varier d'une application à l'autre, même sur différents niveaux de contraintes dans le même problème.

Dans certains cas, il serait possible d'interpoler les fonctions de coût $\mathcal{C}(\boldsymbol{b})$ et la fonction de qualité $\lambda(\boldsymbol{b})$ dans le domaine continu, puis d'appliquer des algorithmes de programmation linéaire ou de programmation non-linéaire, avant de convertir la solution en valeurs

discrètes. On perdra alors la rapidité des algorithmes classiques, comme le glouton par exemple. En effet, la complexité de la conversion de la solution en domaine discret (par l'algorithme de séparation et évaluation, par exemple) est à elle seule en $\mathcal{O}(2^N)$, en gagnant l'optimalité de la solution.

Malgré leur utilisation dans d'autres domaines, certains algorithmes d'optimisation n'ont pas encore été dignement étudiés sur le problème d'optimisation des largeurs de données. Dans les chapitres suivants, nous présentons des améliorations pour les algorithmes génétiques et l'application au problème d'optimisation des largeurs, des algorithmes de recherche avec tabous et GRASP.

Chapitre 5

Améliorations de la technique basées sur les algorithmes génétiques

Les algorithmes génétiques [52] sont des métaheuristiques, permettant d'obtenir une solution approchée en un temps acceptable ou fixé. Ces algorithmes appartiennent à la famille des algorithmes évolutionnistes et utilisent la notion de sélection naturelle développée par Darwin. Tout d'abord, la population est créée à partir de solutions potentielles au problème. Ensuite, le traitement évolutionnaire modifie la population et génère les générations suivantes. Finalement, les individus dominants sont sélectionnés.

Ces algorithmes ont été utilisés plusieurs fois pour l'optimisation des largeurs [66, 4]. Un algorithme génétique multi-objectifs est proposé par Han dans ses travaux de thèse [59]. Cet algorithme a servi de base à nos travaux d'optimisation de la largeur des données basé sur les algorithmes génétiques. Dans ce chapitre, les améliorations proposées visent à obtenir plus rapidement des solutions de meilleures qualités.

5.1 Présentation des algorithmes génétiques

5.1.1 Introduction aux algorithmes évolutionnistes

Les « algorithmes évolutionnistes » représentent une classe des algorithmes basés sur le processus d'évolution naturelle. Un algorithme typique est composé de cinq éléments essentiels :

- une population constituée de plusieurs individus (chaque individu est un candidat pour la solution) ;
- un mécanisme d'évolution composé de plusieurs opérateurs ;
- un mécanisme d'évaluation de l'adaptation de chaque individu ;
- une méthode de sélection permettant d'éliminer certains individus ;
- une population initiale.

La figure 5.1 montre les étapes principales d'un algorithme évolutionniste. Du point de vue opérationnel, un algorithme évolutionniste typique commence par la définition d'une population initiale souvent générée aléatoirement. Ensuite, la phase d'évolution est

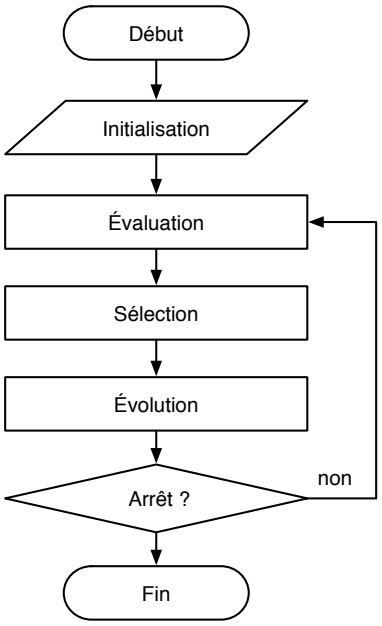


FIGURE 5.1: Graphe flot d'un algorithme évolutionniste

itérative et composée de trois étapes correspondant à l'évaluation de l'adaptation de chaque individu, à la sélection d'une partie des individus, et à l'évolution de la population à partir des individus sélectionnés. Le processus se termine lorsque la condition d'arrêt est vérifiée.

La population initiale est souvent générée aléatoirement, mais dans certaines implémentations de l'algorithme, l'expert peut choisir une population meilleure que les autres. Au cours de l'évolution, les individus les mieux adaptés sont choisis pour reproduire la génération suivante et les moins adaptés sont éliminés. La condition d'arrêt est vérifiée lorsqu'un nombre maximum de générations ou un nombre maximum d'évaluations est atteint.

Représentation d'individus

Dans les algorithmes évolutionnistes, chaque individu est représenté par les allèles dans un chromosome. Un chromosome est souvent codé par une chaîne binaire, mais il existe d'autres codages, dépendant de l'algorithme et du problème. Par exemple, nous verrons par la suite dans le problème d'optimisation des largeurs, où un chromosome est une chaîne d'entiers, chaque entier correspondant à la largeur d'un opérateur. Dans notre exemple, chaque entier est également une configuration de gènes, appelée allèle. Nous utilisons brièvement le terme *chromosome* pour présenter les allèles dans un chromosome et le terme *gène* pour représenter une configuration de gène s'il n'y a pas de confusion.

Généralement, un chromosome ne contient que des informations directement liées au problème. Un changement au niveau des chromosomes se traduit par un changement de l'apparence de l'individu. La partie apparaissant dans l'environnement est le phénotype. Dans la biologie, un chromosome peut contenir des gènes n'ayant aucun effet sur l'individu. Le total des gènes sur un chromosome, effectifs ou non, est le génotype. L'adaptation d'un individu, est souvent une fonction du génotype même si dans la nature, elle devrait être une fonction de phénotype. La distinction entre le génotype et le phénotype est importante

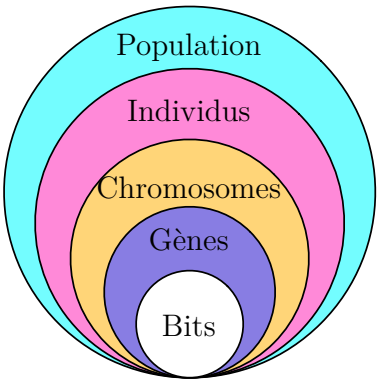


FIGURE 5.2: Notations utilisées pour l'étude des algorithmes évolutionnistes

dans certains algorithmes évolutionnistes. Du point de vue de l'évolution, la connaissance du génotype permet d'avoir la position des gènes et de réaliser les opérations telles que la recombinaison. La mutation, consistant à modifier un point de l'individu, est considérée comme une opération liée au phénotype.

La terminologie génétique est présentée à la figure 5.2. L'unité la plus petite est un ADN (correspondant à un bit). Un gène est une séquence d'ADN. Un chromosome est un porteur de gènes et contient plusieurs gènes. Le nombre de chromosomes des individus d'une même espèce est le même. Dans les algorithmes évolutionnistes, un individu est considéré contenir un seul chromosome. Finalement, la population est un ensemble d'individus.

Fonction d'adaptation

Comme dans tous les problèmes d'optimisation, une fonction d'objectif est définie de la manière suivante :

$$f : A_x \mapsto \mathbb{R} \tag{5.1}$$

où A_x représente l'espace des individus. Cette fonction doit refléter la mesure de la quantité à optimiser et doit avoir une certaine régularité sur l'espace de recherche. De plus, la fonction d'objectif doit permettre de bien différencier les individus. Une fonction ayant quasiment les mêmes valeurs pour tous les individus n'est certainement pas un bon choix.

Le rôle principal d'une fonction d'objectif étant de mesurer la quantité à optimiser, il est parfois difficile de bien distinguer les individus. Ainsi, une fonction d'adaptation est définie séparément de la fonction d'objectif :

$$\Phi : A_x \mapsto \mathbb{R}_+. \tag{5.2}$$

L'ensemble d'arrivée de Φ est non-négatif pour certaines stratégies de sélection, comme la sélection proportionnelle. Dans d'autres cas, $\Phi(a_i(t))$ peut prendre des valeurs dans $\mathbb{R}$. Généralement, la fonction d'adaptation est une combinaison de la fonction d'objectif f et d'une fonction d'échelle (*scaling*) g :

$$\Phi(a_i(t)) = g(f(a_i(t))) \tag{5.3}$$

où $a_i(t) \in A_x$ pour le $i^{\text{ème}}$ individu au temps t. Par exemple, pour minimiser la fonction d'objectif f où la valeur maximale $f_{\max}$ est connue, la fonction d'adaptation peut être définie par

$$\Phi(a_i(t)) = f_{\max} - f(a_i(t)). \tag{5.4}$$

Lorsque la « population évolue », les individus dominants tendent à avoir leurs valeurs d'adaptation dans un intervalle étroit, rendant la sélection plus difficile. Pour résoudre ce problème, différentes méthodes de concentration sur cet intervalle sont utilisées. Grefenstette [54] a proposé une fonction d'adaptation correspondant à une transformation linéaire de la fonction d'objectif et variable au cours du temps. Cette fonction peut être écrite de la manière suivante

$$\Phi(a_i(t)) = \alpha f(a_i(t)) - \beta(t), \tag{5.5}$$

où $\alpha = \pm 1$ en fonction du type de problème (maximisation ou minimisation). Le terme $\beta(t)$ représente la pire valeur de la fonction d'objectif obtenue lors des dernières générations. En général la valeur de $\beta(t)$ s'améliore au cours du temps, ainsi, cette fonction permet d'avoir de meilleures propriétés de sélection. D'autres méthodes sont proposées comme par exemple dans [52, 49].

Mécanismes de sélection

La sélection est un des opérateurs les plus importants dans les algorithmes évolutionnistes. L'objectif de la sélection est de faire ressortir les bons individus. L'opérateur ne crée aucun nouvel individu, il cherche à sélectionner les individus relativement bons et exclure les individus de moins bonnes qualités. Un des attributs du mécanisme de sélection est la force de sélection [114] (*strength of selection* ou *selection pressure*). Celle-ci est un terme informel indiquant la probabilité que les meilleurs individus soient sélectionnés. La force de sélection est mesurée comme le rapport entre l'adaptation maximale et l'adaptation moyenne dans la population.

Les paragraphes suivants présentent les principales méthodes de sélection. Comme dans la suite du chapitre, μ est le nombre d'individus dans d'une population.

a. Sélection proportionnelle Cette stratégie attache à chaque individu une probabilité de reproduction proportionnelle à son adaptation. La probabilité d'un individu i est calculé à l'aide de

$$\Pr_{\text{prop}}(i) = \frac{\Phi(i)}{\sum_{j=1}^{\mu} \Phi(j)}. \tag{5.6}$$

La sélection proportionnelle nécessite que l'adaptation des individus ne soit pas majoritairement située sur un intervalle étroit. Si cette condition n'est pas satisfaite, la sélection devient inefficace et devient une sélection aléatoire.

b. Sélection par tournoi Un tournoi est organisé avec un groupe de q individus aléatoirement sélectionnés dans la population. Le meilleur individu du groupe est gardé pour la

nouvelle génération et ce processus se répète λ fois pour trouver λ individus de la nouvelle génération. L'avantage de cette méthode de sélection est qu'elle n'a pas besoin d'échelle dans la fonction d'adaptation : l'indéterminisme ne joue pas sur l'adaptation, mais sur la sélection pour chaque tournoi. Le deuxième avantage important est la facilité de paralléliser la sélection. En effet, aucun calcul global est effectué.

Le problème du tournoi est la perte de la diversité. Le cas où $q = 1$ (tournoi avec un seul participant) est équivalent à une sélection aléatoire. Plus q est grand, plus la perte est importante. L'expression de la perte de diversité est [12]

$$\theta_{\text{tour}}(q) = q^{-1/(q-1)} - q^{-q/(q-1)}. \tag{5.7}$$

En pratique, la valeur suggérée de q dans plusieurs applications est située dans l'intervalle $[6; 10]$. Dans ce cas, la perte de diversité est de plus de 50%.

c. Sélection par classement Dans ce mécanisme, la probabilité de sélection d'un individu dépend de son classement. Comme dans le tournoi, le problème d'échelle de la fonction d'adaptation n'est pas présent.

Les individus sont classés à travers la valeur associée de la fonction d'adaptation. Ensuite, à chaque individu i, avec le classement $\text{rank}(i)$ est attribuée une valeur $u(\text{rank}(i))$. La probabilité de sélection est proportionnelle à cette valeur :

$$Pr_{\text{rank}}(i) = \frac{u(\text{rank}(i))}{\sum_{j=1}^{\mu} u(\text{rank}(j))} \tag{5.8}$$

Le choix de la suite $u(n)$ définit la variation de classement et peut correspondre à une suite arithmétique (linéaire) ou géométrique.

d. Sélection de Boltzmann La sélection de Boltzmann a été proposé par de la Maza et Tidor [32]. Dans ce cas, la fonction d'adaptation est définie par

$$\Phi(a_i(t)) = \exp \frac{f(a_i(t))}{T}, \tag{5.9}$$

où T est un paramètre assimilé à la température. Ce paramètre permet de modifier la force de sélection. Plus T est grand, moins une différence au niveau de la fonction d'objectif entraine une différence au niveau de la fonction d'adaptation.

Étant donné la forme de la fonction d'adaptation, ce mécanisme est considéré comme une autre méthode de mise à l'échelle de la fonction d'objectif [7].

e. Sélection locale Dans ce mécanisme, la population est divisée en différentes « sous-populations ». La sélection (ainsi que la recombinaison) est effectuée dans chaque sous-population.

f. Sélection par troncature Les individus sont classés par leur adaptation. Seulement les meilleurs individus sont sélectionnés et ils sont considérés égaux pour générer la génération suivante. Alors que les mécanismes de sélection précédents sont plus naturels en intégrant des caractéristiques probabilistes, la sélection par troncature est plus avantageuse pour une large population.

Recombinaison

La recombinaison (enjambement ou croisement) est une opération complexe dans les systèmes biologiques [72]. Les deux chromosomes s'échangent des parties de leurs chaînes, pour donner de nouveaux chromosomes. Ces enjambements peuvent être simples ou multiples.

Dans les algorithmes génétiques, la recombinaison est une opération importante. La probabilité d'apparition d'une recombinaison lors d'un croisement entre deux chromosomes, est entre 0 et 1, et correspond à un paramètre de l'algorithme. La valeur de cette probabilité dépend du problème d'optimisation. Une probabilité trop grande peut entraîner une convergence prématurée.

5.1.2 Algorithmes génétiques

Les algorithmes évolutionnistes ont été développés indépendamment et sont classés en trois grandes familles : stratégies d'évolution, programmation génétique et algorithmes génétiques (AG). L'histoire de ces algorithmes est résumé dans l'annexe A.

Plusieurs étapes pour résoudre un problème avec un algorithme génétique. Une fois que les paramètres sont codés et qu'une fonction d'adaptation est définie, l'algorithme suit les étapes suivantes dont les étapes de 3 à 5 pour le traitement de nouvelles générations se répètent :

1. définition de la première génération ;
2. calcul de la fonction d'adaptation ;
3. sélection ;
4. recombinaison ;
5. mutation.

Codage des paramètres

Contrairement aux stratégies d'évolution et à la programmation évolutionniste, les AG se concentrent sur le génotype. Les AG travaillent sur les chromosomes codés, et nécessitent donc un codage des individus. La fonction d'objectif (5.1) est reécrite à l'aide de l'expression

$$f : A_x \mapsto M \mapsto \mathbb{R}. \tag{5.10}$$

La fonction de décodage $\gamma : A_x \mapsto M$ transforme un chromosome dans une espace spécifique du problème, où la fonction d'objectif est appliqué. La fonction de codage γ^{-1} a pour but de mieux comprendre la représentation des chromosomes.

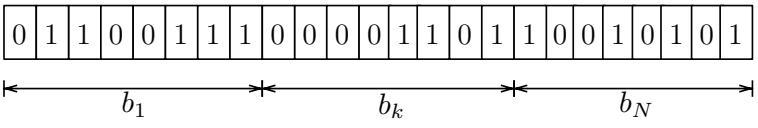


FIGURE 5.3: Codage d'un chromosome dans les algorithmes génétiques

Différents types de codage sont disponibles : valeur entière, valeur réelle, permutation. Chaque codage dispose de différentes techniques de mutation et de recombinaison. Dans notre problème d'optimisation, les paramètres sont codés en binaire (valeur entière). Un chromosome $x = (x_1, x_2, ..., x_n)$ est codé par une chaîne binaire de longueur l égale à

$$l = \sum_{j=1}^{n} l_j, \tag{5.11}$$

où l_j est le nombre maximal de bits pour coder la largeur associée à la variable x_j.

Code Gray Le codage binaire positionnel peut ralentir la convergence d'un algorithme génétique. Ce système de codage n'est pas approprié pour la recombinaison et la mutation. Le problème est illustré à la figure 5.4a. Deux parents sont très proches, mais leurs présentations en simple position sont complètement différentes. La recombinaison avec croisement génère deux enfants très loin de leurs parents, ce qui est un comportement génétique non souhaité. Le même problème est présent dans la mutation. Le changement d'un bit, dépend de la position, peut conduire à une large variation.

Le code Gray est donc souvent utilisé dans les algorithmes génétiques. Ce codage assure qu'un seul bit est différent dans la représentation de deux nombres consécutifs. Autrement dit, la distance de Hamming de deux nombres consécutifs est égale à 1. La figure 5.4b montre un exemple de stabilité de l'utilisation du code Gray.

$\text{parent}_1 =$	(10000000)	$= 128$
$\text{parent}_2 =$	(01111111)	$= 127$
$\text{enfant}_1 =$	(10011111)	$= 159$
$\text{enfant}_2 =$	(01100000)	$= 96$

(a) Codage positionnel

$\text{parent}_1 =$	(11000000)	$= 128$
$\text{parent}_2 =$	(01000000)	$= 127$
$\text{enfant}_1 =$	(01000000)	$= 127$
$\text{enfant}_2 =$	(11000000)	$= 128$

(b) Code Gray

FIGURE 5.4: Le codage position réelle versus le code Gray pour la représentation de chromosome. Le code Gray permet de minimiser la distance Hamming entre les individus proches, de minimiser la perturbation et est souvent meilleur [71] que le codage positionnel dans les algorithmes génétiques.

Détermination de la première génération

Grâce aux caractéristiques de l'AG, la première génération peut être générée de manière aléatoire sans avoir aucun effet sur les générations suivantes. Cependant, une partie de la population peut être générée par une heuristique pour avoir une solution proche de l'optimum. Mais l'utilisation d'heuristiques pour l'initialisation peut réduire la diversité et ainsi conduire à l'obtention de solutions sous-optimales.

Ensuite, l'adaptation de chaque chromosome dans la nouvelle génération est calculée.

Traitement des nouvelles générations

Pour les nouvelles générations, les étapes suivantes sont réalisées.

1. En fonction de l'adaptation de chaque individu, deux individus sont sélectionnés afin de former les parents. Plus l'adaptation d'un individu est grande, plus celui-ci a de chances d'être sélectionné.
2. Ces deux individus sont recombinés avec une certaine probabilité de croisement (50% – 75% généralement).
3. Avec une probabilité très faible (0,1%), des mutations sont réalisées au niveau des progénitures. Cette étape permet d'obtenir de la diversité et d'éviter de converger vers des optima locaux.
4. Ces nouveaux individus sont intégrés dans la nouvelle génération.

Plusieurs études se sont concentrées sur les différentes techniques de recombinaisons. Le rôle des mutations par apport à la recombinaison a été sous-estimé et celui-ci n'a été reconnu que récemment [6].

L'idée de la recombinaison est simple et est reliée au principe du génotype des AG. Étant donnés deux individus, chacun possède des parties ayant de bonnes propriétés présentes dans leurs gènes. Idéalement, un nouvel individu est généré avec la recombinaison des « bonnes parties » de chacun. Mais dans la réalité, ces parties de gène ayant de bonnes propriétés ne sont pas connues (sinon l'optimisation ne serait pas nécessaire). Alors, ces deux individus sont recombinés de façon aléatoire et leur enfants sont observés dans la génération suivante. Parfois un enfant est composé des moins bons blocs que ceux de ses parents, dans ce cas il ne peut pas survivre longtemps dans la processus de sélection.

Condition d'arrêt

La nouvelle génération remplace l'ancienne et son adaptation est calculée. La condition d'arrêt est vérifiée afin de terminer l'algorithme. Dans notre cas, l'algorithme termine après un certain nombre de générations. Dans le cas général, il est difficile de déterminer la condition d'arrêt, parce que les enfants ne sont pas toujours meilleurs que leurs parents.

Il existe généralement différents critères pour la condition d'arrêt :
- l'objectif n'est pas amélioré depuis k générations ;
- l'objectif (ou l'objectif moyen des individus) atteint une limite ;
- le nombre d'évaluations de la fonction d'objectif atteint une limite ;
- le temps d'exécution atteint une limite ;
- tous les individus possibles apparaissent avec un minimum de probabilité [5] ;
- la probabilité d'avoir une amélioration dans la prochaine génération est très faible.

La condition d'arrêt est déterminée par un ou plusieurs critères présentés ci-dessus. Pour les deux premiers critères, le choix nécessite une connaissance approfondie de la fonction d'objectif ou des expérimentations. Pour de nombreux problèmes, il est difficile de décider s'il faut terminer après un nombre donné de générations. Dans ce cas, les deux derniers critères représentent le meilleur choix.

Si la condition n'est pas satisfaite, l'étape précédente est reprise, sinon la solution est sélectionnée dans la génération courante.

5.1.3 Algorithmes génétiques multi-objectifs

Dans notre problème d'optimisation, la condition (4.1) doit être impliquée. Dans ce cas, le problème devient une optimisation à objectifs multiples.

$$\begin{cases} \min\left(\mathcal{C}(\mathbf{b})\right) \\ \min\left(-\lambda(\mathbf{b})\right) \end{cases} . \tag{5.12}$$

Ces deux critères sont contradictoires, ce qui est souvent le cas pour une optimisation à objectifs multiples. Pour un tel problème, soit une fonction agrégée est utilisée pour convertir le problème en mono-objectif [149], soit les solutions sont recherchées grâce à la frontière de Pareto. Dans cette section, tout d'abord, la notion de la frontière de Pareto est présentée. Ensuite, nous trouverons pourquoi une fonction d'agrégation ne peut pas entièrement résoudre le problème. Finalement, les algorithmes basés sur la frontière de Pareto sont présentés.

Frontière de Pareto

En général, un problème d'optimisation multi-objectifs sous contraintes peut être présenté à travers l'expression

$$\min \boldsymbol{f}(x) = (f_1(x), f_2(x), ...f_n(x)) \tag{5.13}$$

sous contrainte de

$$\begin{cases} \boldsymbol{g}(x) = (g_1(x), g_2(x), ...g_m(x)) = \mathbf{0} \\ \boldsymbol{h}(x) = (h_1(x), h_2(x), ...h_p(x)) \geq \mathbf{0} \end{cases} . \tag{5.14}$$

Définition 5.1. *Un individu a domine un individu b si et seulement si*

$$\begin{cases} \forall i \quad f_i(a) \leq f_i(b) \\ \exists j \quad f_j(a) < f_j(b) \end{cases} . \tag{5.15}$$

Si a domine b, on dit également que b *est dominé par* a et ceci est noté $b \prec a$.

Définition 5.2. *Un individu x^* est un optimum de Pareto (ou « Pareto–optimal ») s'il n'est dominé par aucun individu. Autrement dit, $\forall x$:*

$$\begin{bmatrix} \boldsymbol{f}(x) = \boldsymbol{f}(x^*) \\ \exists i \quad f_i(x) > f_i(x^*) \end{bmatrix} . \tag{5.16}$$

Dans cette définition, x^* est un optimum de Pareto s'il n'existe aucune autre solution x qui peut diminuer un critère sans faire augmenter les autres critères. Autrement dit, aucune solution n'est meilleure qu'un optimum de Pareto. Il n'existe pas un seul optimum de Pareto, mais un ensemble de solutions.

Définition 5.3. *L'ensemble des optima de Pareto (des points « non dominés ») forme, par la fonction d'objectif, une frontière appelée frontière d'efficacité de Pareto (ou frontière de Pareto).*

La frontière de Pareto est utilisée dans l'optimisation multi-objectifs. D'autres approches consistent à convertir le problème en mono-objectif. Coello a publié un article de référence [21] sur ce sujet.

Fonction d'agrégation

Cette méthode consiste à recombiner les objectifs par une somme pondérée associant un coefficient de pondération w_i à chaque fonction f_i. L'expression (5.13) devient

$$\min \sum_{i=1}^{n} w_i f_i(x), \tag{5.17}$$

où w_i les coefficients de pondération, correspondant ainsi à une optimisation mono-objectif. Si le but est connu, il est préférable d'utiliser la forme vecteur

$$\min \sum_{i=1}^{n} w_i |f_i(x) - T_i| \tag{5.18}$$

, où $\boldsymbol{T} = (T_1, T_2, ..., T_n)$ est le but à atteindre. Cette dernière forme de conversion en mono–objectif est appelée aussi *programmation par but*.

La méthode de somme pondérée est simple et les solutions trouvées sont non dominées. Cependant, ces solutions varient en fonction des poids w_i. La détermination de ceux-ci est assez délicate. En effet, il n'existe pas de vraie relation entre les objectifs. De plus, alors que la solution est Pareto–optimale (optimale au sens de la frontière de Pareto), l'algorithme ne peut trouver qu'une seule solution.

Algorithme Génétique à Évaluation Vectorielle (VEGA)

Schaffer a proposé l'approche VEGA (pour *Vector Evaluated Genetic Algorithm*) [134]. Dans une génération, n sous–populations de taille N/n (N le nombre total d'individus) sont générées par la sélection proportionnelle à la fonction d'objectif f_i. Ces sous-populations sont ensuite fusionnées en N individus. Les opérations génétiques (recombinaison, mutation) sont appliquées sur ces individus pour avoir la nouvelle génération.

Cette méthode possède des inconvénients. L'algorithme tend à favoriser les individus « spécialisés » correspondant aux meilleurs pour un critère donné. Cependant, le but est de trouver le plus grand nombre d'individus non-dominés. Une des techniques évitant la spécialisation est proposé par Goldberg [52]. Cette technique classe les non-dominés grâce à une connaissance partagée discutée dans la suite.

L'approche VEGA reste aujourd'hui une méthode de référence pour la comparaison des performances entre les algorithmes multi-objectifs.

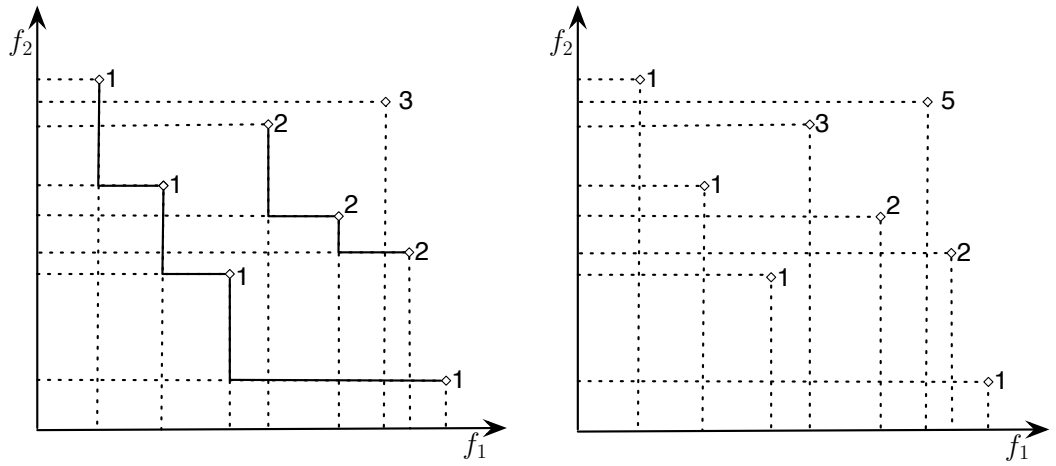
Méthodes basées sur la frontière de Pareto

Afin d'éliminer le phénomène de spécialisation, un classement des individus basé sur la domination est utilisé. Il existe deux méthodes différentes de classement, présentées dans la Figure 5.5. La première méthode, proposée par Goldberg [52], détermine tout d'abord la première frontière de Pareto (F1). Les individus sur F1 sont temporairement ignorés, les nouveaux optima de Pareto, sélectionnés à partir des individus restants, créent la deuxième frontière de Pareto (F2) et ainsi de suite. La deuxième méthode, proposée par

Fonseca [45], détermine les frontières à partir du classement des individus. Un individu x_i dans la génération t dominé par $p_i(t)$ individus possède un classement

$$\text{rank}(x_i(t)) = 1 + p_i(t). \tag{5.19}$$

Tous les individus ayant le même classement sont sur la même frontière de Pareto. La figure 5.5b illustre ce classement. Dans cet exemple, aucun individu est classé 4 ou autrement dit, la frontière F4 ne contient aucun individu. Cette méthode permet de mieux différencier les individus. Par exemple, sur la figure 5.5b, l'individu classé 3 est dominé par deux individus 1 et n'est donc pas sur la même frontière de Pareto avec les deux individus classés 2. Alors que dans la première méthode, ces trois individus ont le même classement.



(a) Méthode 1 (Goldberg, 1989) (b) Méthode 2 (Fonseca, 1993)

FIGURE 5.5: Deux méthodes pour classer les individus sur les frontières de Pareto.

Plusieurs algorithmes sont basés sur ce classement des non dominés. Les algorithmes NSGA (*Nondominated Sorting Genetic Algorithm*) proposé par Srinivas [139] et MOGA (*Multiple Objective Genetic Algorithm*) proposé par Fonseca [45] sont des algorithmes étendus des AG. Dans NSGA, la première version du classement est utilisée alors que MOGA applique la deuxième méthode. Le mécanisme de sélection dans MOGA et celui dans NSGA sont différents. Dans NSGA, avant la sélection, les individus de même classe se voient attribuer la même valeur d'adaptation correspondant à la moyenne des valeurs d'adaptation de tous les individus dans cette classe (adaptation partagée).

Les individus ne sont pas distribués uniformément sur les frontières de Pareto. Afin de garder la diversité, même dans une classe, les individus dans les zones moins peuplées doivent avoir plus de probabilité d'être sélectionnés. Pour pénaliser les groupes d'individus, l'adaptation partagée est alors divisée par un coefficient proportionnel au nombre de voisins ayant la même classe que cet individu. Ce coefficient de voisinage m_i détermine le peuplement d'un individu i et est calculé par

$$m_i = \sum_{j \neq i} \text{Sh}(d(i,j)), \tag{5.20}$$

où $d(i,j)$ est la distance phénotypique (dans l'espace objectif) ou distance génotypique (distance Hamming) entre deux individus i et j, $\text{Sh}(d)$ est une fonction décroissante appelée

fonction de partage. Usuellement $\mathrm{Sh}(d) = 0$ si $d > \sigma_{\mathrm{par}}$ le rayon de voisinage. Dans plusieurs algorithmes, $\mathrm{Sh}(d)$ est défini par

$$\mathrm{Sh}(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{\mathrm{par}}}\right)^{\alpha} & \text{si } d < \sigma_{\mathrm{par}} \\ 0 & \text{sinon} \end{cases}, \tag{5.21}$$

où α et σ_{par} sont des constantes. La constante α est généralement égale à 1 ou 2 et permet de modifier la forme de la fonction de partage. La constante σ_{par} possède un rôle important et détermine la taille des niches.

Dans [33], Deb et al. proposent l'algorithme NSGA–II, une version améliorée de NSGA. Cet algorithme utilise un processus de classement rapide et un estimateur de densité qui permet d'éviter d'utiliser le paramètre σ_{par}. Le concept d'élitisme est également introduit.

Dans [74], Horn et al. proposent l'algorithme NPGA (*Niched Pareto Genetic Algorithm*) basé sur la sélection tournoi. Pour éviter la convergence vers un candidat unique, le tournoi de domination Pareto est utilisé : deux individus sont choisis aléatoirement pour participer au tournoi. Au lieu de les comparer directement entre eux, un jeu de comparaison (*comparison set*) comprenant t_{dom} individus sélectionnés aléatoirement est réalisé. Si un seul des deux individus est dominé par ce jeu de comparaison, l'autre est choisi. Si aucun n'est dominé ou les deux sont dominés, ces individus sont considérés de même classe. Afin de garder la diversité, l'individu avec le plus petit coefficient de voisinage m_i est choisi.

Les autres approches génétiques sont aussi utilisées pour l'optimisation multi-objectif. Knowles et al. [84] proposent l'algorithme PAES (*Pareto Archived Evolution Strategy*). Dans [158], Zitzler propose l'algorithme SPEA (Strength–Pareto Evolutionary Algorithm) utilisant une population externe pour garder les meilleurs individus de toutes les générations. Cet algorithme assure la diversité entre les non-dominés et une implémentation propre permet ainsi de réduire la complexité de calcul.

Une synthèse des techniques utilisées dans ces algorithmes est présentée dans la section 5.3.2. Dans nos travaux, deux techniques, correspondant à l'élitisme et l'ajout d'une recherche locale, sont utilisées pour améliorer l'algorithme MOGA dans le problème d'optimisation des largeurs.

5.1.4 Applications des algorithmes génétiques à l'optimisation des largeurs

Dans le domaine du traitement de signal, les algorithmes génétiques ont d'abord été utilisés dans la conception de filtres numériques. Dans [107], les auteurs proposent d'utiliser un algorithme génétique associé à une descente de gradient. Cette méthode permet d'éviter les optima locaux présents dans les algorithmes gloutons classiques, et d'avoir une meilleure vitesse de convergence que les AG seuls. Différents auteurs [153, 48] utilisent l'algorithme génétique pour chercher les coefficients d'un filtre FIR.

Dans [143], les algorithmes génétiques sont utilisés pour optimiser la largeur des coefficients d'une FFT 16 points pour minimiser la consommation d'énergie dans un récepteur MC–CDMA. La fonction d'adaptation est une somme pondérée du RSBQ (coefficient α) et de la consommation d'énergie (coefficient β). La sélection roulette est utilisée. La valeur β est fixée à 1 lorsque α varie entre 1 et 10 pour trouver la meilleure valeur empirique.

Les probabilités de recombinaison et de mutation sont fixées respectivement à 90% et 10%. Le critère de performance est le RSBQ de référence, obtenu lorsque les largeurs sont uniformes. Des expérimentations sont réalisées pour différents RSBQ correspondant à des largeurs uniformes comprises entre 11 et 16 bits.

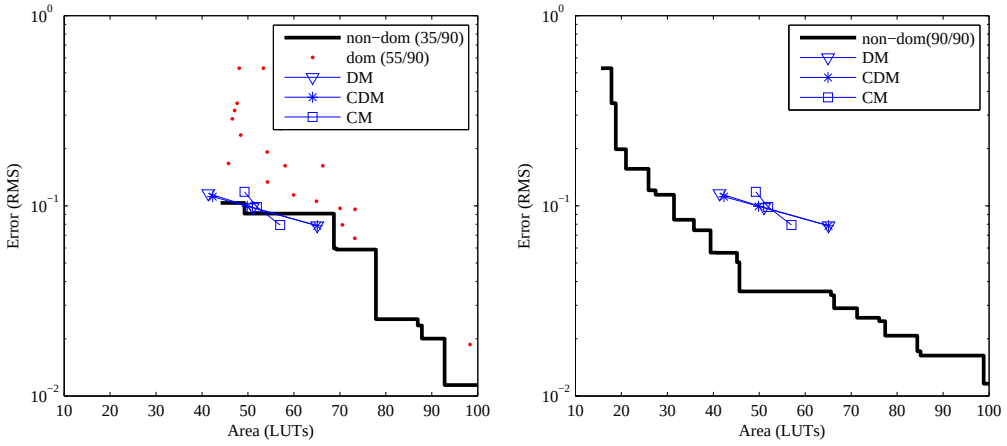
Dans [2], les auteurs de l'Université de Southampton cherchent à optimiser la largeur des données dans le cas de deux équations différentielles d'ordre 10 et 18, un filtre FIR-25 et une DCT 4x4. Trois paramètres : la surface, la consommation et le niveau de bruit sont pris en compte. Un est fixé pour l'optimisation des deux autres. L'algorithme génétique à objectifs multiples avec la technique de somme pondérée est utilisé pour ces optimisations, avec la sélection roulette, les opérations de recombinaison et mutation sont utilisées. Un certain nombre d'individus générés aléatoirement sont introduits dans chaque génération. Le nombre d'individus d'une génération, de recombinaisons, de mutations, d'individus aléatoires ainsi que le nombre de générations sont des paramètres proportionnels au nombre d'opérateurs du système. Les résultats montrent que par rapport aux largeurs uniformes de 8, 16, 24 et 32 bits, les coûts en termes de surface et de consommation d'énergie peuvent diminuer jusqu'à 10%. Cependant les auteurs ne se comparent pas avec les autres algorithmes d'optimisation.

Kyungtae Han de l'Université du Texas à Austin, propose dans sa thèse [59] d'utiliser la frontière de Pareto pour l'optimisation d'un filtre IIR à 7 variables. Dans cette étude, il publie son code MATLAB [60] utilisant la boîte à outils GEATbx avec les paramètres par défaut. Les résultats présentés à la Figure 5.6 montre qu'après 50 générations, le résultat de l'AG est plus ou moins équivalent aux algorithmes glouton classiques. Dans la $500^{\text{ème}}$ génération, l'AG est nettement meilleur que ses concurrents avec un niveau de bruit de 10^{-1}. Il reste pourtant quelques problèmes. Premièrement, même si tous les individus sont sur la frontière de Pareto (à la $500^{\text{ème}}$ génération), cette génération ne garantit pas la meilleure solution. Par exemple, si on cherche une solution avec un niveau de bruit de 10^{-2}, le résultat de la $250^{\text{ème}}$ génération est 10% meilleur que celui de la $500^{\text{ème}}$ génération. Deuxièmement, la frontière de Pareto ne s'améliore pas forcément après une génération. Pour l'optimisation d'un filtre IIR à trois variables, la frontière de Pareto de la $250^{\text{ème}}$ génération est pire que celle obtenue à la $100^{\text{ème}}$ génération. Le temps d'optimisation est assez grand. Pour des exemples à quatre variables, 500 générations n'étaient pas suffisantes pour avoir un bon résultat. Le nombre de générations doit augmenter lorsque le nombre de variables augmente.

5.2 Analyse qualitative des algorithmes génétiques

Le principe des algorithmes génétiques est simple et à pour objectif d'imiter la sélection naturelle. Les paramètres sont codés dans une structure linéaire similaire aux chromosomes. Une population correspond à un ensemble de chromosomes (individus). L'algorithme commence avec une population initiale aléatoire. Des cycles d'évolution perfectionnent les individus.

Les avantages et les inconvénients des algorithmes génétiques dans l'optimisation combinatoire sont présentés dans la suite.



(a) Génération 50e (b) Génération 500e

FIGURE 5.6: Comparaison entre l'algorithme génétique et les recherches locales sur un filtre IIR à 7 variables. Ces frontières Pareto sont obtenues aux 50ème et 500ème générations, sur une population comprenant 90 individus. Les algorithmes gloutons utilisent comme une métrique de sélection de la direction de déplacement la précision (DM), le coût (CM) ou une combinaison des deux (CDM). Source : [59]

5.2.1 Avantages

Parallélisme. Les algorithmes génétiques sont très favorables à la parallélisation pour l'évaluation d'une population. Le calcul de l'adaptation de chaque individu peut facilement être réalisé en parallèle. Les mécanismes de sélection n'utilisant pas d'information globale (par exemple dans le cas de la sélection par tournoi) ceux-ci peuvent aussi d'être parallélisés.

Fonction d'objectif complexe. Les algorithmes génétiques ne sont pas sensibles à la non-régularité, à la non dérivabilité et à la non-continuité de la fonction d'objectif. Les autres algorithmes d'optimisation sont souvent perturbés par la présence d'optima locaux. Même si la fonction d'objectif est dérivable, les AG ne requierent pas d'information sur le gradient de cette fonction.

Multi-objectifs. Dans une optimisation multi-objectifs, il existe plusieurs solutions. Les algorithmes génétiques optimisent sur un ensemble d'individus et peuvent donc facilement s'adapter aux problèmes multi-objectifs.

Une liste de solutions. Les AG ne trouvent pas seulement une seule solution mais une liste de solutions. L'utilisateur a la possibilité de décider laquelle est meilleure, ou, dans certains cas, de retenir toutes les solutions.

Hybridation avec les autres méthodes. Les algorithmes génétiques peuvent utiliser d'autres méthodes d'optimisation, e.g. la recherche locale, pour améliorer les solutions.

Les GA sont aussi le complément des algorithmes classiques pour, par exemple, éviter les optima locaux.

Immunité aux bruits. Dans l'évolution, la mutation est considérée comme une source de bruit. Les algorithmes génétiques utilisent la mutation comme un des opérateurs principaux et sont donc immunisés aux différents bruits dans le génotype et le phénotype.

Problèmes à grande échelle. Les AG sont particulièrement adaptés aux problèmes avec un espace de recherche large.

Éviter les optima locaux. Le nombre d'individus dans la population est identique pour toutes les générations. La recombinaison et la mutation permettent d'avoir toujours de la diversité sur toute la population et ainsi évitent les optima locaux.

5.2.2 Inconvénients

Détermination de la fonction d'adaptation. Dans les AG, la fonction d'adaptation est utilisée au lieu de la fonction d'objectif. La détermination de la fonction d'adaptation est parfois difficile. Ce problème n'est pas présent pour les autres types d'algorithme.

Codage des variables. Le type de codage et la taille des gènes dépendent du problème. Pour plusieurs problèmes, différents types de codage sont possibles. Le choix du codage est un paramètre déterminant pour l'obtention de bonnes performances. Cependant, le nombre de codages différents est limité.

Détermination des paramètres. La détermination de certains paramètres n'est pas aisée. Plusieurs paramètres doivent être déterminés : la taille de la population, le taux de mutation, le taux de recombinaison et le mécanisme de sélection.

Impossibilité d'utilisation du gradient. Les algorithmes basés sur le calcul du gradient peuvent rapidement trouver la solution d'un problème d'optimisation sur les fonctions analytiques convexes et simples. Les algorithmes génétiques sont moins efficaces dans ce cas parce que les informations sur la fonction d'objectif ne sont pas utilisées.

Convergences prématurées. La convergence prématurée est un des principaux obstacles dans les problèmes complexes et de grande taille. Les algorithmes génétiques classiques convergent parfois vers des optima locaux. Rudolphe a même prouvé [131] que la version d'origine [52] des GA ne convergent jamais vers l'optimum global.

Difficulté dans la détermination des optima exacts. Les algorithmes génétiques recherchent sur des points séparés dans une espace large et il est donc difficile de trouver les optima exacts mais les AG fournissent des individus proches des optima.

Condition d'arrêt. Il existe différents critères pour déterminer la condition d'arrêt des algorithmes génétiques. Malheureusement, peu de fondements théoriques ont été développés sur ces aspects.

Nombre d'évaluations de la fonction d'adaptation important. Plus la taille du problème est grande, plus le nombre d'individus dans la population est grand et plus le nombre de générations est grand pour une convergence vers les optima de Pareto. Cela nécessite un volume de calcul important et donc des temps d'optimisation élevés.

5.2.3 Conclusion

Dans la réalité, les AG sont largement utilisés même si, pour de nombreux problèmes, les fonctions d'objectif sont très complexes à formuler. Les algorithmes génétiques, avec leurs multiples variations, sont souvent plus performants que les algorithmes traditionnels dans les problèmes difficiles. Les recherches sur les AG cherchent à développer les bases théoriques. Pour ne citer que quelques unes, Harik propose dans [64] un modèle permettant de prédire la qualité de la convergence en se basant sur plusieurs paramètres : la taille de la population, la taille du problème ainsi que la difficulté du problème [1]. Dans [89], une approche probabiliste des AG permettant de trouver une bonne solution en un temps très court est proposée. Cette approche, dénommée EDA (*Estimation of Distribution Algorithms*), est prometteuse et rencontre un certain succès dans l'optimisation combinatoire. Cependant, elle requiert des techniques complexes qui dans un premier temps ne seront pas utilisées dans ces travaux de thèse.

Dans la partie suivante, nous proposons des techniques simples pour améliorer l'utilisation des algorithmes génétiques pour le problème d'optimisation des largeurs.

5.3 Proposition d'un algorithme amélioré

5.3.1 Contexte

Un algorithme basé sur MOGA pour l'optimisation des largeurs est présenté dans [59, 60]. Avec un nombre de générations suffisamment grand, les résultats montrent qu'il est plus performant que les algorithmes gloutons classiques. Cet algorithme a été implémenté au sein d'une boîte à outil MATLAB : GEATbx.

L'algorithme utilise le codage en entier pour les largeurs. Deux objectifs à minimiser sont le niveau de bruit de quantification et le coût (la surface ou la consommation d'énergie). La force de sélection est fixée à une valeur entre 1 et 2. Le mécanisme de sélection est réalisé par classement non-linéaire. Dans la nouvelle génération, les nouveaux individus ne représentent que 90% de la génération. 10% restant sont des copies de certains meilleurs parents. Ce paramètre, présent dans GEATbx, est une implémentation de l'élitisme.

Dans la suite, nous présentons les techniques issues de la littérature utilisées pour améliorer les approches existantes d'optimisation des largeurs basées sur les algorithmes

1. Dans [64], la difficulté d'un problème est quantifié et dépend de différents paramètres y compris le nombre de gènes, la variance des adaptations des individus et la distance entre les adaptations de deux meilleurs individus.

génétiques. Certaines de ces techniques sont ensuite appliquées dans notre proposition d'algorithme génétique pour l'optimisation des largeurs dans la section 5.3.3. Les performances sont comparées avec les résultats récents dans la section 5.3.4.

5.3.2 Extensions des algorithmes génétiques

Ces techniques avancées peuvent être classées dans deux catégories : les techniques générales liées à l'évolution biologique et les techniques demandant la connaissance de la fonction d'objectif.

Techniques générales

Élitisme L'objectif de l'élitisme est de garantir que la meilleure adaptation ne diminue pas dans la prochaine génération. Différentes implémentations de cette technique sont disponibles. Par exemple, les meilleurs individus peuvent être copiés pour la prochaine génération s'ils n'avaient pas été sélectionnés.

Phénomène de spéciation et niche La convergence prématurée est la situation dans laquelle tous les individus d'une génération sont identiques ou quasiment identiques, et se positionnent dans une solution sous optimale. Dans ce cas, les mutations et recombinaisons aident très peu pour la suite de l'évolution. L'utilisation d'une population plus large réduit la probabilité de cette situation, mais le nombre de calculs est nettement augmenté.

En réalité, les individus tendent vers la spéciation. Ce phénomène se manifeste pour les individus ayant des points communs. Ils séparent la communauté, se regroupent et créent une nouvelle espèce. Cela est une part de l'évolution biologique et peut être implémentée dans les algorithmes génétiques.

Différentes techniques sont employées. Par exemple, les individus se trouvant dans une niche partagent leur adaptation : l'adaptation d'un individu est divisée par un coefficient correspondant au nombre d'individus dans la niche. Une niche avec de très bons individus ne peut donc pas majorer la population et permet aussi d'assurer la diversité et d'éviter la convergence prématurée. Une autre technique simple consiste à ne pas permettre la recombinaison sur deux individus parents dans deux espèces parce qu'il promeut la création des enfants non viables. Cette technique est particulièrement utile dans les problèmes où la fonction d'adaptation est composée de plusieurs pics et la moyenne sur deux pics se situe souvent sur une valeur non souhaitée.

Connaissance de la fonction d'objectif

La plupart des améliorations se concentre sur les opérateurs généraux, cependant certaines cherchent à améliorer l'algorithme pour les problèmes pour lesquels la fonction d'objectif est connue. Les chercheurs n'utilisent pas ces connaissances pour améliorer les opérations d'échange comme la recombinaison, mais pour trouver les bonnes directions de déplacement. Par exemple, un individu ne respectant pas la contrainte sera exclu sans évaluer la fonction d'objectif.

En ayant connaissance du gradient ou de la monotonie de la fonction d'objectif, plusieurs extensions vont utiliser la recherche locale comme un outil d'amélioration de la performance des algorithmes génétiques [52, 30, 104]. Dans un schéma hybride, la recherche locale permet d'avoir une solution de bonne qualité après que les AG aient convergé vers une solution proche de l'optimal.

Ainsi, avec la connaissance des directions où se trouvent les optima, la mutation devient sélective.

5.3.3 Proposition d'une version améliorée de MOGA

Différentes techniques d'amélioration des AG présentées dans la section 5.3.2 ont été intégrées à l'implémentation présente dans les paragraphes précédents. Ces extensions permettent de ne pas modifier le cœur de l'algorithme génétique. Les techniques retenues sont l'élitisme et la recherche locale (algorithme hybride).

Élitisme

Dans la littérature, l'élitisme consiste à réinsérer les « élites » dans la nouvelle génération. Cette technique est déjà présente dans GEATbx. Un pourcentage de meilleurs parents est en compétition pour l'intégration dans la nouvelle génération. Cependant, nous voulons néanmoins conserver exactement les individus souhaités et modifier le mécanisme de sélection. Cette technique ne peut pas être réalisée avec GEATbx. Pourtant, l'élitisme est considéré comme l'une des meilleures techniques permettant d'améliorer les algorithmes génétiques. Ainsi, dans un premier temps, nous limitons la mise en œuvre de l'élitisme à deux approches correspondant à l'élitisme total et au semi-élitisme. Ces approches utilisent une mémoire externe pour garder les élites et ne modifient pas la population.

a. Élitisme total Cette technique consiste à conserver la frontière de Pareto dans toutes les générations. Pour cela, nous injectons dans la fonction d'objectif un mécanisme d'extraction de tous les individus de la frontière de Pareto de la génération actuelle dans une population externe. Cette population est gérée afin de ne conserver seulement que les optima de Pareto. Ce principe est présenté dans l'algorithme 5.1. Les lignes 2–4 sont ajoutées. La population externe, elites(t), nécessite l'utilisation d'une mémoire externe avec une taille dépendant du nombre de générations.

Algorithme 5.1 Élitisme total par une génération externe dans MOGA

Entrées: Génération actuelle : $a_i(t)$, $1 \leq i \leq \mu$
Sorties: Objectif de chaque individu, *plus* une population externe elites(t) contenant tous les optima de Pareto des générations 1 à t

1: calculer $f(a_i(t))$, $1 \leq i \leq \mu$
2: best(t) $\leftarrow$ les optima Pareto dans $a_i(t)$ *injection*
3: elites(t) $\leftarrow$ les optima Pareto dans (best(t) $\cup$ elites($t-1$)) *dans la*
4: mémoriser elites(t) dans la mémoire *fonction d'objectif*
5: **return** $f(a_i(t))$

La complexité additionnelle de cette technique est liée au calcul des optima de Pareto. Dans les problèmes où le temps d'évaluation de la fonction d'objectif est important comme dans le cas de l'évaluation de la précision par simulation, ce calcul est négligeable. Dans les autres cas, la complexité de l'algorithme réside dans le mécanisme de sélection où se passent plusieurs tris et tirages au sort.

Cependant, il est possible d'optimiser le calcul des optima Pareto. Remarquons que si t est grand, le nombre d'individus dans best(t) est négligeable par rapport au nombre d'individus dans elites($t-1$). Nous pouvons envisager un algorithme permettant de vérifier chaque individu dans best(t) s'il est rejeté ou accepté dans elites(t). Si cet individu est accepté, alors les individus dans elites($t-1$) devant être supprimés seront recherchés. De plus, best(t) étant une frontière de Pareto, il existe probablement des techniques de comparaison plus efficaces à mettre en œuvre. Dans cette thèse, nous n'avons pas étudié plus profondément les techniques d'amélioration de l'implémentation.

b. Semi-élitisme Un des plus grands inconvénients de l'élitisme total est la taille de la mémoire et la complexité de calcul. Une technique plus simple permettant d'améliorer l'algorithme MOGA permettant d'avoir un compromis entre la mémoire et la complexité est proposée.

Le problème d'optimisation des largeurs est différent des problèmes MOGA généraux. En effet, les optima de Pareto non favorables sont connus a priori. Ce sont les individus insatisfaisant le critère de performance correspondant à la contrainte de précision. Ainsi, si nous avons besoin du résultat pour une contrainte de précision donnée, la seule valeur intéressante dans chaque génération correspond à l'individu sur la frontière de Pareto ayant le coût minimal et satisfaisant le critère de performance. Seul cet individu est conservé et est comparé entre les générations. Lorsque l'algorithme s'arrête, le résultat obtenu correspond au meilleur individu de toutes les générations.

Algorithme 5.2 Semi-élitisme dans MOGA

Entrées: Génération actuelle : $a_i(t)$, $1 \leq i \leq \mu$
Sorties: Objectif de chaque individu, *plus* une elite(t) – le meilleur individu des générations 1 à t

1: calculer $f(a_i(t))$, $1 \leq i \leq \mu$
2: best$(t) \leftarrow \arg\min f_2(a_i(t))$ s.c. $f_1(a_i(t)) \leq f_{1,\text{req}}$
3: elite$(t) \leftarrow$ meilleur(best(t), elite$(t-1)$)
4: mémoriser elite(t)
5: **return** $f(a_i(t))$

L'algorithme 5.2 est quasiment identique à l'algorithme 5.1, excepté qu'un seul individu de la frontière de Pareto est retenu et une seule élite est conservée ce qui permet de diminuer la complexité.

Recherche locale

La recherche locale permet d'améliorer la solution trouvée par l'algorithme génétique. La recherche locale est effectuée avec un algorithme glouton max-1 parce que l'individu trouvé satisfait déjà le critère de performance. Nous utilisons le rapport entre les gradients

sur la précision et le coût comme critère de sélection de la meilleure direction de déplacement dans l'algorithme glouton (voir 4.2.1). Ce critère évalue l'apport sur la performance par unité de coût et est plus pertinent que les critères basés sur la précision ou le coût ou la CDM (*Cost Distortion Measure*).

La complexité de l'algorithme est négligeable par apport à celle de l'algorithme génétique. Nous pouvons donc modifier l'algorithme pour qu'il soit plus performant avec un compromis acceptable sur la complexité.

Il est utile de remarquer qu'il est possible d'améliorer la solution par un algorithme de recherche plus performant présenté dans la section 6.1.

5.3.4 Expérimentations

Fonction de coût Le coût de l'implantation est évalué à travers la consommation d'énergie. Une bibliothèque d'opérateurs contenant la consommation d'énergie sur une cible ASIC est utilisée pour les simulations. Les caractéristiques de ces opérateurs se trouvent dans l'annexe B. Le coût $\mathcal{C}$ correspond à la consommation d'énergie de l'ensemble des opérateurs

$$E = \sum_{i=1}^{N} (E_{\mathrm{op}_i} \times \mathrm{nb}_{\mathrm{op}_i}), \tag{5.22}$$

avec E_{op_i} la consommation d'énergie de $i^{\text{ème}}$ opérateur et $\mathrm{nb}_{\mathrm{op}_i}$ le nombre d'opérations effectuées par cet opérateur.

Paramètres de simulation La taille de la population est fixée à 90 et le nombre de générations est fixé 500. Cette valeur est obtenue empiriquement[59] : tous les individus sont non–dominés pour les problèmes de taille petite et les résultats sont assez bons pour les problèmes de taille plus grande.

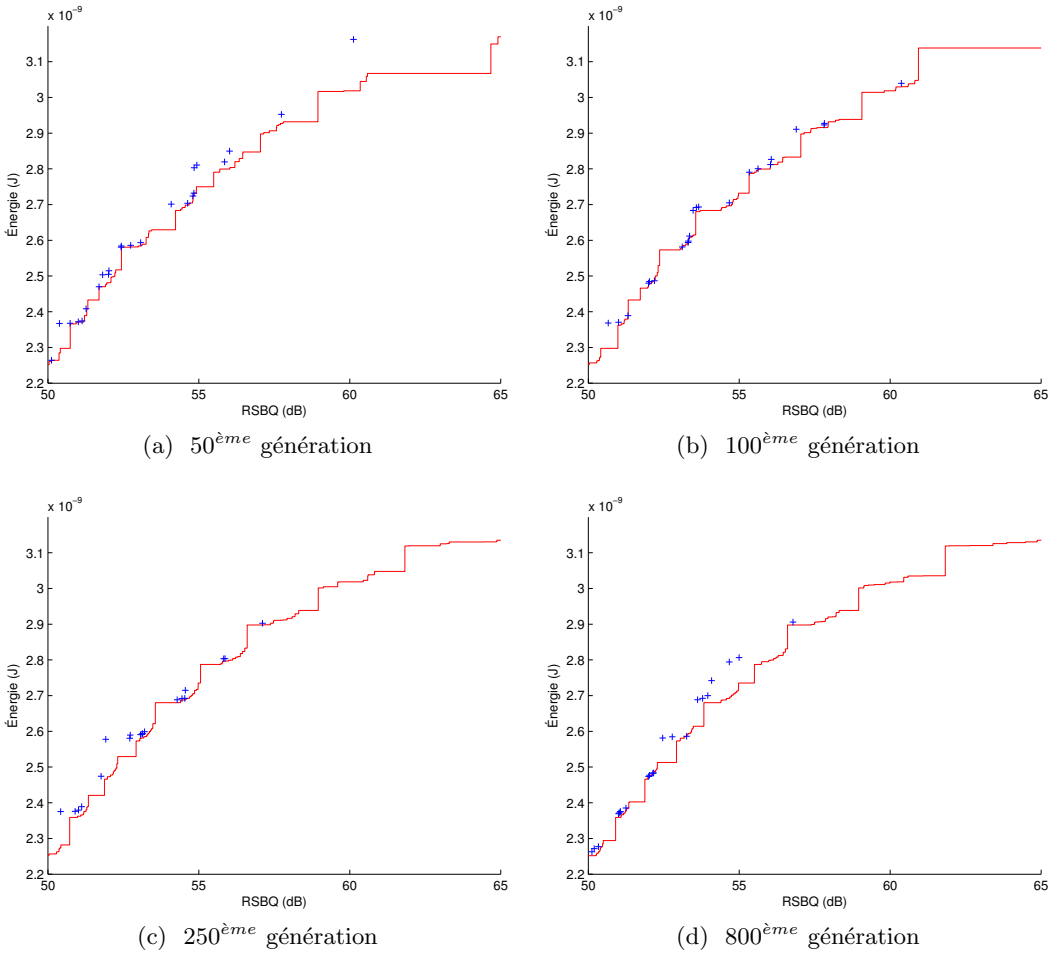
Quatre sous–populations sont utilisées, chacune possède son propre taux de mutation. La compétition entre les sous–populations permet de transférer des individus vers de meilleures sous–populations. 20% d'individus sont migrés, c-à-d ajouté puis supprimé, entre les sous–populations.

Les résultats dans cette section sont obtenus par simulation sur un Macbook Pro Core 2 Duo T7400 2,16 GHz, 4 GB RAM (PC1).

L'effet de l'élitisme

Dans un premier temps, nous vérifions, aux travers de différentes expérimentations, l'efficacité de l'élitisme dans le cas où tous les individus sur la frontière de Pareto sont conservés pour la prochaine génération.

Les résultats obtenus pour un filtre IIR d'ordre 8 (IIR–8), composé de 18 variables, sont présentés à la figure 5.7. L'axe des abscisses représente la précision à travers le RSBQ et l'axe des ordonnées la valeur du coût de la solution. Ce coût correspondant à la consommation d'énergie est exprimé en Joules. L'élitisme (points rouges reliés) permet d'améliorer nettement le résultat. Alors que les résultats sans élitisme (points bleus) à la $50^{\text{ème}}$ sont assez éloignés des meilleurs solutions, l'élitisme donne une frontière de Pareto très proche



(a) $50^{ème}$ génération (b) $100^{ème}$ génération

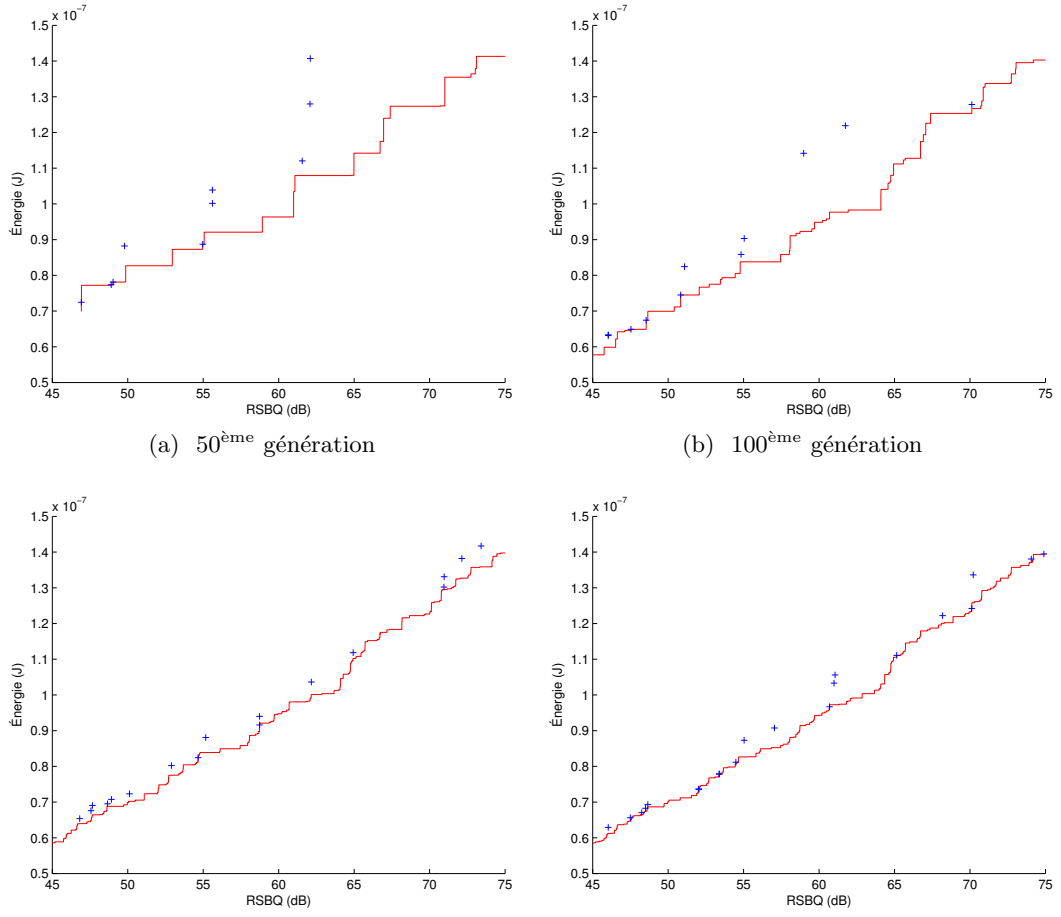
(c) $250^{ème}$ génération (d) $800^{ème}$ génération

FIGURE 5.7: Frontières de Pareto obtenues pour un filtre IIR composé de 18 variables. Les résultats sont présentés pour différentes générations (machine : PC1).

de celles obtenues à la $250^{\text{ème}}$ et $800^{\text{ème}}$ génération sans l'élitisme. En plus, l'élitisme permet de garder un nombre important d'individus sur toutes les valeurs de chaque critère. Par exemple, dans la $800^{\text{ème}}$ génération, sans élitisme il n'y a aucun individu présent sur l'intervalle de $RSBQ$ de 55 dB à 57 dB. Cela signifie que si le critère est à 55 dB, la solution avec le coût minimal est celle pour un RSBQ plus élevé et égal à 57 dB.

Les résultats obtenus pour une FFT sur 64 points composée de 8 variables et un filtre NLMS 128 points composé de 49 variables sont présentés aux figures 5.8 et 5.9. Pour la FFT, le résultat est plus clair : avec l'élitisme, la frontière de Pareto à la $50^{\text{ème}}$ génération est très proche de celle obtenue à la $500^{\text{ème}}$ génération sans élitisme. Sur le filtre NLMS, bien que les frontières de Pareto dans les deux cas sont proches, l'élitisme permet toujours d'étendre la présence des individus sur un intervalle plus large.

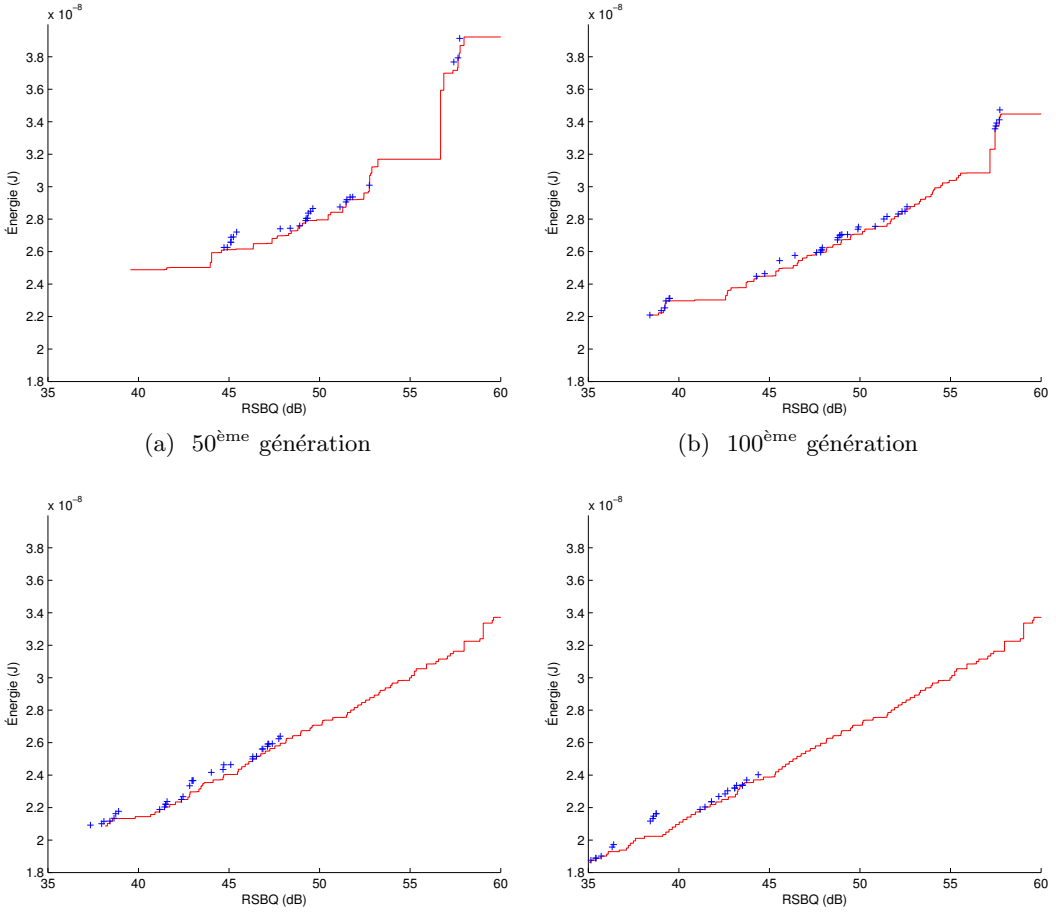
Cette technique réalise un compromis entre le nombre de générations et la taille de la mémoire. Dans le cas du filtre IIR 18 variables, sur une population de 90 individus, l'ensemble des élites contient 287 individus à la $100^{\text{ème}}$ génération et 2363 individus à la $800^{\text{ème}}$ génération. Lorsque la taille de mémoire est proportionnelle au nombre de générations n, le coût de stockage est proportionnel à n^2.



(a) 50ème génération (b) 100ème génération

(c) 250ème génération (d) 500ème génération

FIGURE 5.8: Efficacité de l'élitisme par rapport au nombre de générations. Le résultat est plus marqué quand le nombre de générations est faible. Le nombre d'individus sur la frontière de Pareto augmente naturellement avec le nombre de générations. Les résultats sont présentés pour une FFT–64 composée de 8 variables à optimiser(machine : PC1).



(a) 50ème génération (b) 100ème génération

(c) 250ème génération (d) 500ème génération

FIGURE 5.9: Efficacité de l'élitisme. Simulation sur un filtre NLMS 128 entrées avec 49 variables. La conservation totale des élites permet d'augmenter la diversité et d'avoir une meilleure solution dans certains intervalles de précision : p. ex. à la 500ème génération, pour un RSBQ de 37 dB à 42 dB (machine : PC1).

Semi-élitisme et recherche locale

Dans cette section, nous comparons les performances de notre proposition (MOGA+) avec un algorithme génétique multi-objectifs MOGA classique (MOGA). Les deux algorithmes génétiques sont couplés et l'amélioration correspond à un module intégré dans le processus d'optimisation. Sachant que le temps d'exécution de la recherche locale est négligeable devant le temps d'exécution global pour une centaine de générations, nous obtenons une complexité proche de celle de MOGA.

Les résultats de l'algorithme MOGA et notre MOGA+ sont présentés dans les tableaux 5.1, 5.2 et 5.3 pour un IIR d'ordre 8 composé de 18 et 36 variables et un filtre NLMS. n_g représente le nombre de générations. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité. L'utilisation du semi-élitisme et de la recherche locale permet d'améliorer significativement la qualité de la solution obtenue. Pour 10 générations, l'approche proposée permet d'améliorer pour les trois expérimentations entre 16 % et 48 %. Pour 500 générations, l'amélioration varie entre 8 % et 25 %. La solution obtenue après 10 générations est de bonne qualité. En effet, l'utilisation de 500 générations ne permet d'améliorer que de 1 % à 8 % la solution obtenue avec 10 générations dans le filtre IIR. Pour le NLMS de 18 variables, la solution trouvée par MOGA+ à la $1000^{\text{ème}}$ génération n'est pas différente de celle trouvée à la $500^{\text{ème}}$ génération.

TABLE 5.1: Comparaison de deux algorithmes génétiques sur un filtre IIR–8 (18 variables). n_g représente le nombre de générations. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	n_g	Temps (s)	$n_{\mathcal{C}}$	n_λ	Énergie (J)
MOGA	10	975	804	2496	$3,6580.10^{-9}$
MOGA+	10				$2,6427.10^{-9}$
MOGA	500	19248	39252	39806	$2,7283.10^{-9}$
MOGA+	500				$2,4402.10^{-9}$

TABLE 5.2: Comparaison de deux algorithmes génétiques sur un filtre IIR–8 (36 variables). n_g représente le nombre de générations. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	n_g	Temps (s)	$n_{\mathcal{C}}$	n_λ	Énergie (J)
MOGA	10	3491	804	8170	$3,1758.10^{-9}$
MOGA+	10				$1,6254.10^{-9}$
MOGA	500	20206	39522	42517	$2,1630.10^{-9}$
MOGA+	500				$1,6004.10^{-9}$

Conclusion

L'élitisme permet de diminuer nettement le nombre de générations, donc le temps d'optimisation. La première méthode mémorise toute la frontière de Pareto permettant

TABLE 5.3: Comparaison de deux algorithmes génétiques sur un filtre NLMS 128 entrées, (49 variables). n_g représente le nombre de générations. $n_{\mathcal{C}}$ et n_λ sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	n_g	Temps (s)	$n_{\mathcal{C}}$	n_λ	Énergie (J)
MOGA	500	4001	40753	61212	$2,59.10^{-8}$
MOGA+	500				$2,15.10^{-8}$
MOGA	1000	7478	80753	101212	$2,34.10^{-8}$
MOGA+	1000				$2,15.10^{-8}$

d'améliorer cette frontière et est donc intéressante pour les problèmes d'optimisation sur différentes contraintes. La deuxième méthode possède un surcoût négligeable lié à la recherche locale. Cette méthode est intéressante pour les problèmes d'optimisation des largeurs de données classiques, réalisés pour une contrainte de précision donnée.

5.4 Conclusions

Dans ce chapitre, les algorithmes génétiques ont été utilisés pour optimiser la largeur des données. Les améliorations proposées par rapport aux approches existantes ont été détaillées. Les résultats des expérimentations montrent l'intérêt de l'élitisme et du couplage de la recherche locale pour améliorer la qualité de la solution et le temps d'obtention de celle-ci. Certaines techniques n'ont pas pu être implémentées et vérifiées en raison de l'utilisation de la boîte à outil GEATbx. Cet outil possède aussi des limites sur le nombre de générations, qui est nettement inférieur à la limite de la mémoire. Les outils libres, comme l'infrastructure ParadisEO–MOEO [91], sont des candidats pour les futures expérimentations. Avec ces outils, d'autres techniques, comme par exemple la mutation vers des buts, peuvent être considérées.

Les AG sont des algorithmes d'optimisation avancés permettant d'éviter les optima locaux. Dans la plupart des cas, les AG sont capables de trouver une solution de bonne qualité avec un nombre raisonnable d'itérations. Un autre avantage des AG est la possibilité de paralléliser l'algorithme.

Cependant, les AG ne permettent pas d'utiliser d'autres propriétés du problème que la fonction d'adaptation. Les AG sont efficaces en les combinant avec d'autres algorithmes, par exemple une recherche locale à partir de la solution, ou une recherche locale au fur à mesure des itérations. Cette dernière idée est utilisée au sein des algorithmes stochastiques présentés dans le chapitre suivant.

Chapitre 6

Recherche locale stochastique

Dans ce chapitre, l'optimisation de la largeur des données est réalisée avec un algorithme de recherche locale stochastique. Tout d'abord, une recherche locale déterministe, couplée à un algorithme glouton et à une recherche avec tabous est présentée et appliquée à notre problème d'optimisation. Ensuite, nous présentons les approches de recherche locale stochastique, dont GRASP. Finalement, une approche utilisant GRASP est utilisée dans l'optimisation des largeurs. La recherche avec tabous fait partie de cet algorithme.

6.1 Recherche avec tabous

6.1.1 Introduction

La recherche avec tabous [50, 51] est une métaheuristique d'optimisation devant être utilisée en coopération avec d'autres méthodes. Le couplage de l'algorithme glouton et de la recherche avec tabous peut permettre d'obtenir une meilleure recherche locale et ainsi d'éviter des déplacements inutiles.

Le détail d'une recherche avec tabous est présenté à l'algorithme 6.1. Soit x un candidat dans l'espace de recherche X. Notons s un déplacement, $S(x)$ est l'ensemble des déplacements possibles à partir de x. $S(x)$ peut être considéré comme la fonction de voisinage de x. À l'étape k, un déplacement s_k optimal et absent de la liste des tabous (définie ci-après) est choisi. Après chaque étape, la meilleure solution x^* et la liste des tabous T sont mises à jour. L'algorithme termine lorsque la condition d'arrêt est satisfaite : soit aucun déplacement n'est possible, soit le nombre d'itérations atteint la limite.

En fait, la liste des tabous est une mémoire permettant de modifier le voisinage afin qu'il ne contienne pas de candidats non souhaités. Une des stratégies est de mémoriser tous les candidats explorés afin de ne pas tomber dans une boucle infinie. Dans le cas où la taille de la mémoire, implantant la liste des tabous, est inférieure au nombre de candidats, cette liste des tabous peut être implémentée sous la forme d'un file FIFO.

Algorithme 6.1 Recherche avec tabous (algorithme général)

```
x ∈ X                                   x : état actuel (candidat), X : espace de recherche
x* ← x                                  x* : meilleure solution
T ← ∅                                   T : liste des tabous
k ← 0                                   itération k
tantque S(x) − T ≠ ∅ et k < k_max faire  condition d'arrêt
  k ← k + 1
  choisir le meilleur déplacement s_k ∈ S(x) − T
  x ← s_k(x)
  si C(x) < C(x*) alors
    x* ← x
  fin si
  mettre à jour T
fin tantque
```

6.1.2 Application au problème d'optimisation des largeurs

Métrique d'amélioration de précision ∇_k

La recherche avec tabous est basée sur un algorithme glouton. Il est nécessaire d'avoir une métrique f_{dir} pour choisir la direction de déplacement. Le choix de la direction dépend d'une métrique ∇_k. Cette valeur peut mesurer l'amélioration de la précision après chaque déplacement [15]. Dans ce cas, la métrique ∇_k, correspondant au gradient sur la précision, est définie de la manière suivante :

$$\nabla_{k/\lambda} = \frac{\lambda(\boldsymbol{b}^{\text{next}}) - \lambda(\boldsymbol{b})}{|\boldsymbol{b}^{\text{next}} - \boldsymbol{b}|} \tag{6.1}$$

où $\boldsymbol{b}^{\text{next}}$ est une fonction de $(\boldsymbol{b}, k, \text{direction})$ déterminant la position suivante de $\boldsymbol{b}$ pour le $k^{\text{ième}}$ opérateur. Cette métrique permet de sélectionner la direction fournissant la meilleure précision mais l'augmentation de coût n'est pas prise en compte. Afin de choisir le meilleur compromis en termes de coût et de précision, la métrique ∇_k est définie à partir des gradients sur la précision et le coût à l'aide de la relation suivante

$$\nabla_k = \nabla_{k/\lambda\mathcal{C}} = \frac{\nabla_{k/\lambda}}{\nabla_{k/\mathcal{C}}} = \frac{\lambda(\boldsymbol{b}^{\text{next}}) - \lambda(\boldsymbol{b})}{\mathcal{C}(\boldsymbol{b}^{\text{next}}) - \mathcal{C}(\boldsymbol{b})}. \tag{6.2}$$

Avec ces métriques, les algorithmes gloutons sont classifiés en *glouton–a* (pour *accuracy*, utilisant la métrique $\nabla_{k/\lambda}$) et *glouton–ac* (pour *accuracy/cost*, utilisant la métrique $\nabla_{k/\lambda\mathcal{C}}$. L'algorithme min+1 est un algorithme glouton–a.

Présentation de l'algorithme

L'algorithme proposé pour améliorer les algorithmes gloutons est présenté à l'algorithme 6.1. Cet algorithme combine la recherche avec tabous et les algorithmes gloutons min+1 et max–1. La métrique définie à l'équation 6.2 est utilisée pour sélectionner la direction de déplacement. L'objectif de l'algorithme glouton max–1 est de descendre jusqu'à la

limite $\lambda_{\min}$ puis de s'arrêter. Dans notre cas, l'exploration est poursuivie en espérant trouver une meilleure solution. Lorsque la limite $\lambda_{\min}$ est dépassée, la variable considérée est marquée puis la direction de recherche est inversée. Ensuite, l'algorithme se poursuit tant que la limite n'est pas dépassée, puis marque l'opérateur et inverse la direction et réitère pour chaque opérateur. Le processus s'arrête lorsque tous les opérateurs sont marqués.

Les lignes 1–3 initialisent l'algorithme. La liste des tabous T représente l'ensemble des opérateurs marqués et à ne plus utiliser. Le terme *direction* indique le sens de la recherche. Celui-ci est ascendant (min+1) ou descendant (max–1). Le couple *bestCost* et *bestWL* représente la meilleure solution obtenue en termes de coût et les largeurs associées. La partie principale de l'algorithme est itérative tant que les opérateurs ne sont pas tous marqués (ajoutés dans la liste des tabous).

Les lignes 6 à 12 permettent pour chaque opérateur de vérifier si un déplacement est possible. Si un opérateur atteint la limite ($b_k^{\max}$ dans le sens montant et $b_k^{\min}$ dans le sens descendant), il est ajouté à la liste des tabous. Si un déplacement est possible, la métrique d'amélioration de précision ∇_k de ce déplacement est calculée aux lignes 13–18. Le coût et la précision de chaque position envisagée sont également comparés avec le meilleur résultat actuel afin de ne manquer aucun candidat.

Après le calcul de ∇_k pour chaque opérateur, les lignes 20–22 permettent de vérifier si la liste des tabous est déjà complète. Si c'est le cas, l'algorithme s'arrête avec la solution actuelle.

Dans la dernière partie de l'algorithme, l'opérateur ayant la valeur de ∇_k la plus élevée est choisi si direction > 0, ou avec la plus petite la valeur de ∇_k dans le cas contraire. Cela permet d'avoir la meilleure efficacité précision/coût lorsque l'objectif est d'augmenter la précision. Dans la direction descendante, l'objectif est de diminuer le coût tout en conservant une diminution de précision la plus faible, ainsi l'opérateur ayant une valeur de ∇_k minimale est sélectionné.

Chaque fois que la précision dépasse la contrainte fixée, la direction est inversée. Finalement, afin d'éviter de boucler, l'opérateur choisi est marqué et ne sera plus testé par la suite.

Remarques

Remarque 6.1. *La qualité de la solution obtenue avec l'algorithme 6.2 est identique ou meilleure que celle obtenue avec l'algorithme glouton classique.*

Démonstration. Un algorithme glouton se termine dès que la frontière associée à la contrainte est atteinte. L'algorithme proposé continue la recherche et, après chaque étape, conserve ou améliore la solution. Le résultat trouvé est donc identique ou meilleur que celui trouvé par un algorithme glouton. □

Remarque 6.2. *L'algorithme 6.2 se termine après un nombre fini d'itérations.*

Démonstration. Si la liste des tabous T contient tous les opérateurs, alors l'algorithme se termine.

Algorithme 6.2 Recherche avec tabous

Entrées: solution $\boldsymbol{b}$
Sorties: meilleure solution que $\boldsymbol{b}$

1: $T \leftarrow \emptyset$ *les opérateurs déjà dépassés*
2: direction $\leftarrow (\lambda(\boldsymbol{b}) > \lambda_{\min})\ ?\ -1 : 1$ *sélectionner la bonne direction*
3: bestCost $\leftarrow \infty$; bestWL $\leftarrow \infty$
4: **tantque** $|T| < N$ **faire**
5: **pour tout** $1 \leq k \notin T \leq N$ **faire** *calculer le gradient*
6: **si** direction $> 0 \wedge \boldsymbol{b}_k < \boldsymbol{b}_k^{\max}$ **alors**
7: $\boldsymbol{b}^{\text{cur}} \leftarrow \text{succ}(\boldsymbol{b}, k)$
8: **sinon si** direction $< 0 \wedge \boldsymbol{b}_k > \boldsymbol{b}_k^{\min}$ **alors**
9: $\boldsymbol{b}^{\text{cur}} \leftarrow \text{pred}(\boldsymbol{b}, k)$
10: **sinon**
11: $T \leftarrow T \cup \{k\}$
12: **fin si**
13: **si** $k \notin T$ **alors**
14: $\nabla_k \leftarrow f_{\text{dir}}(\boldsymbol{b}^{\text{cur}}, \boldsymbol{b})$ *calcul de la métrique de choix du déplacement*
15: **si** $\lambda(\boldsymbol{b}^{\text{cur}}) > \lambda_{\min}$ **alors**
16: si nécessaire, mettre à jour bestCost, bestWL
17: **fin si**
18: **fin si**
19: **fin pour**
20: **si** $|T| = N$ **alors** *il ne reste plus aucun essai*
21: arrêter
22: **fin si**
23: **si** direction > 0 **alors** *prendre le meilleur choix et continuer*
24: $j \leftarrow \arg\max \nabla_k$
25: $\boldsymbol{b} \leftarrow \text{succ}(\boldsymbol{b}, j)$
26: **si** $\lambda(\boldsymbol{b}) > \lambda_{\min}$ **alors**
27: direction $\leftarrow -1$
28: $T \leftarrow T \cup \{j\}$
29: **fin si**
30: **sinon**
31: $j \leftarrow \arg\min \nabla_k$
32: $\boldsymbol{b} \leftarrow \text{pred}(\boldsymbol{b}, j)$
33: **si** $\lambda(\boldsymbol{b}) < \lambda_{\min}$ **alors**
34: direction $\leftarrow 1$
35: **fin si**
36: **fin si**
37: **fin tantque**
38: **return** bestWL

Lorsque l'algorithme débute avec une solution satisfaisant le critère de précision, après chaque étape, soit la somme des largeurs $\sum b_i$ décroît, soit la taille de la liste des tabous T augmente. Parce que la somme des largeurs est bornée par 0, la taille de T doit augmenter.

Si l'algorithme commence avec une valeur ne satisfaisant pas le critère de précision, la somme des largeurs est augmentée jusqu'à ce qu'une solution soit trouvée, ou la taille de T doit augmenter. Cette situation revient à celle du premier cas.

Dans tous les cas, la taille de T augmente après un nombre limite d'itérations. L'algorithme s'arrête lorsque cette valeur atteint N. □

6.1.3 Expérimentations et résultats

Les performances en termes de qualité et de temps d'optimisation de cet algorithme de recherche avec tabous sont comparées avec celles obtenues par l'algorithme glouton. Les résultats sur un filtre IIR d'ordre 8 composé de 4 cellules d'ordre 2, sur une FFT de différentes tailles (32, 64 et 128 points) et sur un filtre adaptatif NLMS (128 points) sont présentés dans le tableau 6.1. La première colonne représente l'application et la seconde le nombre de variables N. Afin de faire varier le nombre de variables au sein de ce problème d'optimisation, différentes assignations des opérations aux opérateurs sont réalisées. Ensuite, la taille des opérateurs est optimisée. Sur les différents tests, l'algorithme améliore les résultats du glouton. Dans certains cas, l'algorithme de recherche avec tabous permet de diviser le coût de l'implantation par un facteur 3. L'augmentation du temps d'exécution est relativement importante (50% à 200%), mais ce temps d'optimisation reste raisonnable car les algorithmes d'optimisation gloutons sont rapides.

Les résultats dans cette section sont obtenus par simulations sur un Dell, 2x Xeon 3,2 GHz, 6 GB RAM (PC2).

Les résultats présentés dans le tableau 6.1 sont présentés pour un seul critère de précision. Afin de mieux mesurer la performance des algorithmes, les simulations sont ensuite réalisées avec différents critères de précision. Les résultats sont illustrés aux figures 6.1, 6.2 et 6.3. Le coût moyen de la solution et son écart-type sont présentés. Une large différence entre les solutions en fonction du critère de précision est observée, particulièrement dans les cas des FFT 8 et 12 variables et dans le cas des NLMS. Cependant, nous retrouvons la même tendance que celle obtenue dans le tableau 6.1. Sur un IIR 18 variables, l'amélioration moyenne est environ 15 % au lieu de $6,6$ % pour le critère mesuré dans la table 6.1.

6.1.4 Conclusions

La recherche avec tabous correspond à une modification peu complexe des algorithmes gloutons mais nous avons montré théoriquement et à travers des expérimentations que les performances sont meilleures que celles des algorithmes gloutons. Dans certains cas, lorsqu'un algorithme glouton se trompe de direction, la recherche avec tabous peut augmenter largement la qualité de la solution. Dans la partie suivante, nous présentons une autre technique permettant d'améliorer la solution en utilisant des recherches stochastiques.

TABLE 6.1: Comparaison de l'algorithme glouton et de la recherche avec tabous sur les résultats et le temps d'exécution. Trois applications IIR, FFT et NLMS sont configurées sur différents nombres de variables (machine : PC2).

		Consommation d'énergie (10^{-9}J)			Temps d'optimisation (s)		
Application	N	Glouton	Tabou	Amélioration	Glouton	Tabou	Surcoût
IIR	9	2,1746	2,1746	0%	2,10	6,23	199 %
IIR	13	2,0779	2,0340	2,11%	3,33	8,93	168 %
IIR	14	1,9863	1,9280	2,94%	4,58	14,63	219 %
IIR	18	3,0735	2,8702	6,62%	35,1	83,2	137 %
IIR	36	1,5389	1,4372	6,61%	78,3	177,0	126 %
FFT	4	21,312	21,312	0%	0,32	0,48	49,5 %
FFT	8	397,51	136,26	65,7%	26,0	62,8	141,5 %
FFT	12	331,87	124,89	62,4%	57,3	163,4	185,0 %
FFT	20	282,37	280,85	0,54%	57,4	128,8	124,6 %
FFT	28	1587,0	1575,4	0,73%	2843	4404	54,9 %
NLMS	7	39,013	39,013	0%	67,5	123,3	82,6 %
NLMS	10	12,231	12,231	0%	21,36	57,65	169,9 %
NLMS	13	27,835	24,347	12,5%	168,8	371,5	120,0 %
NLMS	18	13,686	13,600	0,63%	74,6	227,5	205,0 %
NLMS	25	22,776	19,068	16,3%	765,6	1524	99,0 %
NLMS	49	20,186	18,238	9,65%	2865	5796	102,3 %

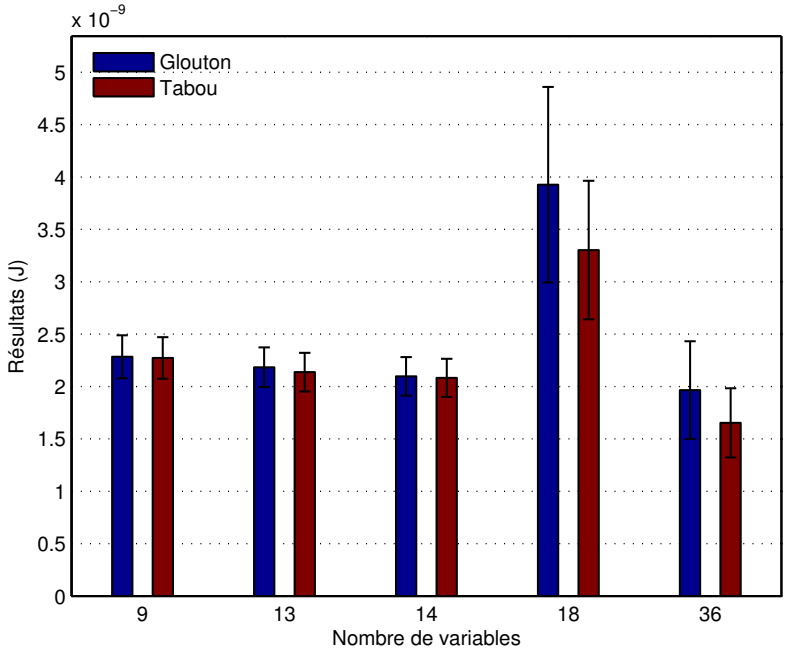


FIGURE 6.1: Comparaison des performances de l'algorithme glouton et de l'algorithme de recherche avec tabous sur les filtres IIR. L'écart-type représente la variation du résultat en fonction du critère de précision (machine : PC2).

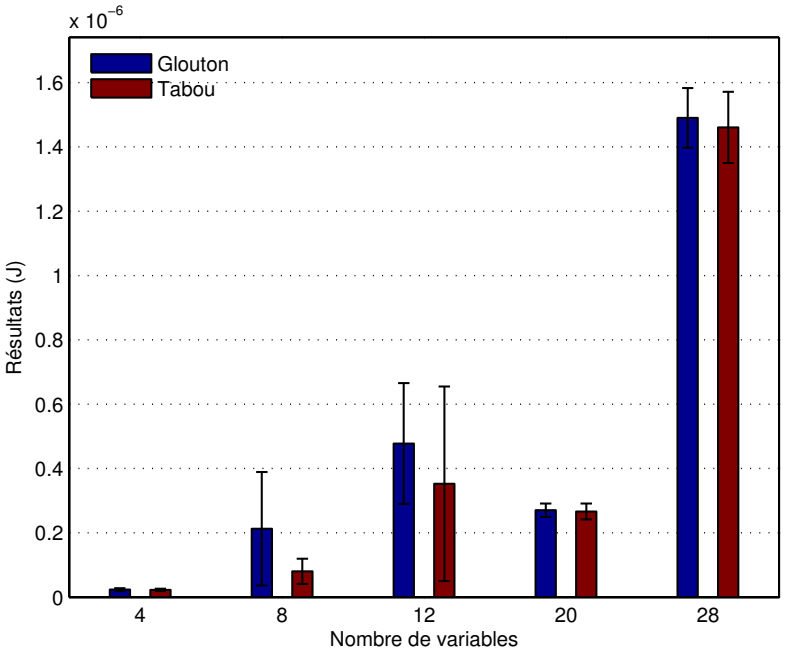


FIGURE 6.2: Comparaison des performances de l'algorithme glouton et de l'algorithme de recherche avec tabous sur les FFT. L'écart-type représente la variation du résultat en fonction du critère de précision (machine : PC2).

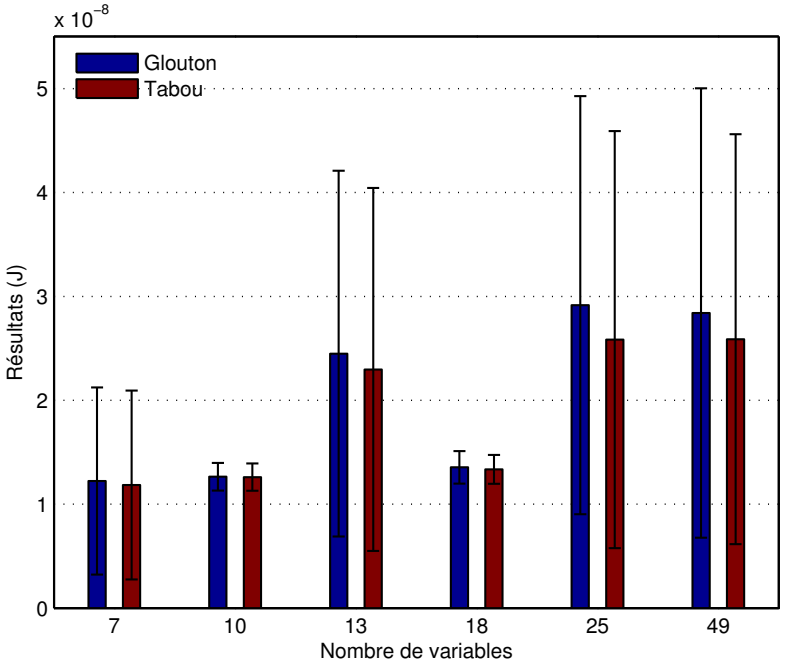


FIGURE 6.3: Comparaison des performances de l'algorithme glouton et de l'algorithme de recherche avec tabous sur les filtres NLMS. L'écart-type représente la variation du résultat en fonction du critère de précision (machine : PC2).

6.2 Procédures de recherche gloutonne aléatoire et adaptative (GRASP)

6.2.1 Algorithmes de recherche locale stochastique

Les algorithmes simples de recherche locale comme le glouton ou la recherche avec tabous permettent de trouver rapidement un optimum local mais la solution est souvent relativement éloignée de la solution optimale. *De plus, une nouvelle exécution de l'algorithme ne permettra pas d'obtenir une solution meilleure.* Afin de contourner ces problèmes, plusieurs études ont été menées sur les algorithmes de recherche locale stochastiques.

Dans le cadre d'une recherche locale, un point de départ est défini et à chaque étape, l'algorithme se déplace vers le meilleur voisin selon les informations locales. Une amélioration envisageable est d'augmenter la taille du voisinage. Soit k *la distance des voisins* (voir section 4.2.1 pour rappel), un k–optimal est un optimum par rapport à tous les voisins ayant une distance égale ou inférieure à k. Un optimum global est un k–optimal pour toutes valeurs de k. Pour un problème à n variables, la taille du voisinage est $O(n^k)$, i.e. exponentielle avec k. Théoriquement, un voisinage de taille idéale possède tous les voisins optimaux comme optima globaux. Il n'est pas réaliste de construire un voisinage très large. Dans le cas de problèmes de taille (n) importante, l'utilisation du voisinage quadratique $(k = 2)$ ou cubique $(k = 3)$ peut conduire à un temps de calcul prohibitif. Afin de réduire la taille du voisinage, différentes méthodes [154, 113, 105] ont été proposées. Même si certaines modifications peuvent avoir des gains de performance considérables, nous nous intéressons par la suite à une autre méthode plus répandue : l'approche stochastique.

Dans le cadre d'une recherche stochastique, la sélection du point de départ ainsi que celle du voisin est aléatoire. De plus, l'algorithme peut disposer d'une mémoire limitée pour stocker les solutions examinées. Ces algorithmes sont développés depuis 1951 [128] pour les problèmes d'optimisation classiques. En 1973, S. Lin utilise l'approche stochastique dans un problème d'optimisation combinatoire [93] et ces algorithmes sont encore largement étudiés actuellement. Parmi les algorithmes proposés, les plus simples utilisent un point de départ aléatoire (à chaque nouvelle itération), ou de temps en temps prennent un mauvais voisin (*Randomised Iterative Improvement* - RII) afin d'éviter les optimums locaux. Une alternative à cette méthode est de ne pas choisir aléatoirement un voisin, mais en fonction de sa qualité. Plus un voisin est de qualité, plus sa sélection sera probable. Cet algorithme est appelé « amélioration itérative probabiliste » (*Probabilistic Iterative Improvement* - PII). La fonction de probabilité joue alors un rôle très important. Supposons s la solution actuelle et s' un de ses voisins (choisi aléatoirement avec une distribution uniforme). Si le critère de Métropolis [103] est utilisé, la probabilité que s' soit sélectionné est déterminée par

$$P(s, s', T) = \begin{cases} 1 & \text{si } f(s') \leq f(s) \\ e^{\dfrac{f(s) - f(s')}{T}} & \text{sinon} \end{cases} . \tag{6.3}$$

Ce critère est un cas particulier du recuit simulé (algorithme 4.1). Plus le nombre d'itérations utilisées pour trouver la solution est élevé, meilleure sera la solution. Cependant, il est difficile de quantifier a priori le nombre d'itérations nécessaires pour obtenir une bonne solution. Ces différents problèmes ont entraîné l'apparition de l'algorithme de recherche

avide aléatoire adaptative (GRASP pour *Greedy Randomized Adaptive Search Procedures*). Cet algorithme permet de trouver une bonne solution en un temps raisonnable.

6.2.2 Algorithme de recherche avide aléatoire adaptative

Les procédures de recherche avide aléatoire adaptative GRASP ont été introduites en 1989, par Feo et Resende [42]. Les étapes d'un algorithme GRASP sont présentées à l'aide de l'algorithme 6.3. Dans la première étape (phase de construction), une solution est *itérativement construite en ajoutant les éléments dans chaque itération*. Ces éléments sont pris au hasard à partir d'une liste obtenue par l'algorithme glouton. Dans la seconde étape, une recherche locale est utilisée pour améliorer la solution. Ces deux étapes sont répétées. Finalement l'algorithme retourne la meilleure solution trouvée.

Algorithme 6.3 GRASP : procédure générale

```
Initialisation du problème
tantque condition d'arrêt non satisfaite faire
  solution ← solution glouton aléatoire                  phase de construction
  solution ← recherche locale (solution)                 phase de recherche locale
  mettre à jour (solution, meilleure solution)           mise à jour de solution
fin tantque
```

Phase de construction

L'efficacité d'une recherche locale dépend de plusieurs critères comme la structure de voisinage, le choix de la direction de déplacement et le point de départ. La structure de voisinage est définie par la distance entre les solutions. Le choix de direction est déterminé par la meilleure direction en fonction du coût et de la précision. La phase de construction joue un rôle important dans le choix du point de départ. Le but est de construire une bonne solution pour la recherche locale, mais aussi de garder assez de diversité afin de ne pas être bloqué dans un optimum local.

La phase de construction est présentée dans l'algorithme 6.4. À chaque itération, la fonction gloutonne trouve tous les candidats, les meilleurs sont mis dans une liste RCL (« *Restricted Candidate List* »). Un candidat de la liste est choisi aléatoirement et ajouté à la solution.

Algorithme 6.4 GRASP : procédure générale – phase de construction

```
Solution ← ∅
tantque solution non complète faire
  Génération de RCL (liste des candidats)          partie de l'algorithme glouton
  s ← random(RCL)
  Solution ← Solution +s
  Mettre à jour la fonction gloutonne en fonction de s          partie adaptée
fin tantque
```

Avec ce nouveau candidat, la fonction gloutonne est réévaluée. Cette partie de l'algorithme est similaire au demi-glouton [65], mais en général, la solution trouvée n'est pas

optimale, l'algorithme est suivi par une recherche locale. Les résultats empiriques montrent que cette phase améliore largement la solution avec peu de calculs supplémentaires.

La taille de RCL est soit fixée par une valeur fixe ou en fonction de la taille de la solution, soit adaptée en fonction de qualité des candidats dans chaque itération. Dans le dernier cas, un candidat s est ajouté dans la liste si

$$f(s) \leq f_{\min} + \alpha \cdot (f_{\max} - f_{\min}) \tag{6.4}$$

avec α un coefficient entre $[0, 1]$. La valeur $\alpha = 0$ correspond au cas d'un algorithme glouton et $\alpha = 1$ correspond à une recherche aléatoire.

Phase de recherche locale

Dans la phase de recherche locale, un algorithme glouton de type recherche avec tabous présenté dans la section 6.1 est utilisé pour obtenir de meilleures performances. La solution non-optimale dans la phase de construction est utilisée comme le point de départ pour la recherche. Lorsque cette phase est terminée, la solution trouvée est le résultat de cette itération. Si la condition d'arrêt, dans notre cas, le nombre d'itérations, n'est pas encore satisfaite, l'algorithme GRASP effectue une nouvelle itération. Dans le cas contraire, la meilleure solution de toutes les itérations est retenue comme résultat de l'algorithme.

6.2.3 Avantages et inconvénients

L'algorithme GRASP, en combinant un algorithme glouton avec un algorithme aléatoire, permet de bénéficier des atouts de ces deux approches. Le glouton aléatoire donne une bonne variété des solutions testées. La meilleure solution trouvée par un algorithme glouton aléatoire est meilleure que celle obtenue par un algorithme glouton. Par rapport à l'algorithme de recherche aléatoire, à partir de la solution trouvée, la recherche locale requiert moins d'étapes pour trouver une très bonne solution.

L'algorithme GRASP est composé de deux phases séparées et ainsi son implantation est facilitée par le développement de deux algorithmes indépendamment. Par apport aux autres métaheuristiques, GRASP possède un seul paramètre important correspondant à la taille de la liste RCL. Ainsi, le paramétrage de l'algorithme est facilité. Mais le principal inconvénient de GRASP réside aussi dans ce paramètre. Globalement, la meilleure valeur de la taille de RCL n'est pas encore connue. La valeur permettant d'avoir un compromis entre le temps de calcul et la qualité de solution [126] est utilisée. Cette valeur doit non seulement fournir une grande variété de solutions à chaque itération mais doit aussi permettre d'avoir un niveau élevé de solutions trouvées.

La qualité de la meilleure solution est améliorée après chaque itération. Cependant, la probabilité d'amélioration décroît en fonction du nombre d'itérations. Malgré cette décroissance, augmenter le nombre d'itérations est la seule méthode pour améliorer la qualité. Comme le temps de calcul ne varie pas beaucoup d'une itération à l'autre, le temps d'optimisation peut être facilement estimé en fonction du nombre d'itérations.

6.3 Application de GRASP à l'optimisation des largeurs

Nous proposons à travers l'algorithme 6.5, une application de la méthode GRASP pour notre problème d'optimisation des largeurs des données. Toutes les itérations commencent à partir d'un même point de départ $\boldsymbol{b}^{\min}$. Dans chaque itération, un algorithme glouton min+1 (Algorithme 4.3), avec modification du critère de sélection de candidat, est effectué. Puis une recherche locale trouve l'optimum local. Cette solution est comparée à la meilleure solution trouvée et la remplace si celle-ci est meilleure. Les détails de l'algorithme sont présentés par la suite.

Algorithme 6.5 GRASP : application dans l'optimisation des largeurs

$\boldsymbol{b} \leftarrow \boldsymbol{b}^{\min}$
tantque condition d'arrêt non satisfaite **faire**
  $\boldsymbol{b}^g \leftarrow$ solution glouton aléatoire
  Recherche locale de $\boldsymbol{b}^g$
  $\boldsymbol{b} \leftarrow$ meilleure$(\boldsymbol{b}, \boldsymbol{b}^g)$ *mise à jour de solution*
fin tantque

6.3.1 Phase de construction

La première étape est la phase de construction. Cette partie de l'algorithme est présentée à travers l'algorithme 6.6.

Algorithme 6.6 GRASP : phase de construction

$\boldsymbol{b} \leftarrow \boldsymbol{b}^{\min}$
tantque RSBQ$(\boldsymbol{b}) <$ RSBQ$_{\min}$ **faire**
  pour $k = 1$ à N **faire**
    $q_k \leftarrow \nabla_k(\boldsymbol{b})$
  fin pour
  RCL $\leftarrow T_{\text{RCL}}$ meilleures valeurs de q_k
  $(q, i) \leftarrow$ valeur aléatoire dans RCL
  $\boldsymbol{b} \leftarrow$ succ$(\boldsymbol{b}, i)$
fin tantque

Dans cette étape, la largeur de chaque opérateur est augmentée d'un bit en gardant pour les autres opérateurs leur ancienne valeur. La liste RCL contient les opérateurs ayant conduit aux valeurs de ∇_k les plus élevées. La métrique ∇_k présentée dans la section 6.1.2 est soit le critère de précision, soit le critère de précision/coût. Si un opérateur atteint sa limite supérieure et ne peut pas être augmenté, sa métrique ∇_k est considérée à $-\infty$ et cet opérateur n'est jamais choisi.

Un opérateur est choisi aléatoirement dans cette liste comme candidat pour l'itération suivante, jusqu'à ce que le candidat trouvé satisfasse le critère de précision : RSBQ$(\boldsymbol{b}) \geq$ RSBQ$_{\min}$.

6.3.2 Phase de recherche locale

Dans la phase de recherche locale, un algorithme de recherche avec tabous est utilisé. Cet algorithme est démontré plus performant qu'un algorithme glouton. Cette phase est similaire à l'algorithme 6.2, avec comme point de départ la solution trouvée dans la phase de construction.

6.3.3 Discussion

Dans les phases de construction et de recherche locale, nous avons deux choix pour la métrique ∇_k. Ces deux choix correspondent à deux variantes de l'algorithme. La première correspondant à l'utilisation de $\nabla_{k/\lambda}$ est appelée *GRASP–a* (pour *accuracy*). La deuxième variante correspondant à l'utilisation de $\nabla_{k/\lambda\mathcal{C}}$ est appelée *GRASP–ac* (pour *accuracy/cost*). Même si la deuxième variante est en général plus performante que la première, ceci n'est pas vrai dans tous les cas. Les algorithmes GRASP–a et GRASP–ac peuvent être considérés comme des améliorations des algorithme glouton–a et glouton–ac en introduisant de la diversité dans la recherche. Nous préférons par la suite garder ces deux variantes et la meilleure solution sera retenue.

Dans nos expérimentations, la condition d'arrêt est le nombre d'itérations. Cette valeur est choisie telle que le temps d'exécution de GRASP soit comparable à celui des algorithmes utilisés dans la littérature.

6.4 Évaluation de GRASP pour l'optimisation des largeurs

Afin de comparer les algorithmes, différentes applications de traitement du signal ont été testées. Dans le cadre du récepteur WCDMA, pour le module de recherche de trajets et le décodeur, le nombre de variables à optimiser est faible. Pour ces deux applications, la meilleure solution obtenue est très proche du point de départ correspondant à la combinaison des largeurs minimales, les solutions trouvées par tous les algorithmes sont identiques. Ainsi, des applications avec un nombre plus important de variables à optimiser, ont été testées. Pour ces expérimentations, les applications utilisées sont un filtre à réponse impulsionnelle infinie (IIR) d'ordre 8 composé de 4 cellules d'ordre 2, une FFT sur 64 points et un filtre adaptatif NLMS sur 128 points. Le nombre de variables à optimiser est de 14, 18 et 36 pour le filtre IIR, 8, 12, 20, 28 pour la FFT et 7, 10, 13, 18, 25, 49 pour le filtre NLMS. Ces différentes valeurs sont obtenues en jouant sur l'affectation des opérations aux opérateurs. Pour ces expérimentations, la contrainte de précision (puissance du bruit de quantification) est fixée à une valeur de -55 dB, -46 dB et -40 dB respectivement pour le filtre IIR, la FFT et le filtre NLMS.

Les algorithmes d'optimisation des largeurs « classiques » comme le glouton–a, glouton–ac et la CDM (*Cost–Distortion Mesure*), ainsi que les algorithmes génétiques et GRASP sont évalués et présentés dans les tables 6.2 à 6.7. Le premier algorithme, dénommé glouton–a, correspond à l'algorithme min+1 classique. L'algorithme glouton–ac effectue une recherche locale ayant $\nabla_{k/\lambda\mathcal{C}}$ comme critère d'évaluation des candidats potentiels. L'algorithme CDM proposé par Han [58] est également une recherche locale avec le critère $\alpha \cdot \triangle\mathcal{C} + (1-\alpha) \cdot \triangle\lambda$ ($\alpha > 0$). Dans cette expérimentation le paramètre α est fixé à 0,5, et correspond probablement à la meilleure valeur [58, 61].

L'algorithme génétique MOGA [59] et sa version avancée MOGA+ proposée dans la section 5.3 sont évalués. La version avancée nécessite très peu de calculs supplémentaires, permettant ainsi de l'implanter directement dans l'algorithme d'origine. Dans toutes les applications, les algorithmes génétiques s'arrêtent à la $10^{\text{ème}}$ génération et à la $500^{\text{ème}}$ génération. Le nombre d'itérations de GRASP est fixé à 10 pour avoir un résultat et un temps d'optimisation comparable aux algorithmes génétiques.

La nature aléatoire des algorithmes génétiques et de GRASP nécessite d'exécuter plusieurs fois l'algorithme [1], puis le résultat moyen est utilisé pour l'analyse. La taille de la liste RCL est fixée à 2 dans les deux cas. Le temps de calcul est mesuré en secondes. Le nombre d'évaluations de la fonction de coût et de la fonction de qualité tient compte des doublons. Si une combinaison des largeurs est évaluée plusieurs fois, seule la première fois est comptée parce que la mémoire cache utilisée est suffisamment grande (uniquement la clé de hachage 32 bits de données et le résultat sont stockés). Les algorithmes probabilistes et itératifs comme AG et GRASP exécutent le même calcul plusieurs fois.

6.4.1 Algorithme GRASP–a

Dans un premier temps, l'algorithme GRASP–a basé sur le critère $\nabla_{k/\lambda}$ est comparé avec les autres. Les résultats du filtre IIR sont détaillés dans le tableau 6.2 pour un problème avec 18 variables et dans le tableau 6.3 pour un problème avec 36 variables. Pour le filtre IIR 18 variables, parmi les algorithmes déterministes, le CDM trouve la meilleure solution. Le glouton–ac est un peu moins performant mais avec un temps d'exécution plus faible. L'algorithme glouton–a, même en utilisant le même temps d'exécution que le CDM, donne la pire solution. Quant aux algorithmes stochastiques, les algorithmes génétiques ne fournissent pas de bons résultats avec seulement 10 générations comme nous avons observé dans le chapitre 5. L'algorithme Génétique+ est bien meilleur et à la $500^{\text{ème}}$ génération, il est meilleur que tous les algorithmes déterministes. Mais la meilleure solution est celle de GRASP–a à sa $10^{\text{ème}}$ itération avec un temps d'exécution deux fois moins important que les algorithmes génétiques à leur $500^{\text{ème}}$ génération.

Table 6.2: Comparaison de différents algorithmes d'optimisation sur un filtre IIR–8 (18 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	Temps (sec)	$n_{\mathcal{C}}$	n_{λ}	Énergie (J)
Glouton–a	1452	1	2050	$3,9197 \times 10^{-9}$
Glouton–ac	649	793	883	$2,6054 \times 10^{-9}$
CDM [58]	1425	1827	1917	$2,4750 \times 10^{-9}$
MOGA (10)	975	804	2496	$3,6580 \times 10^{-9}$
MOGA+ (10)	.	.	.	$2,6427 \times 10^{-9}$
MOGA (500)	19248	39252	39806	$2,7283 \times 10^{-9}$
MOGA+ (500)	.	.	.	$2,4402 \times 10^{-9}$
GRASP–a (10)	9381	1	22847	$2,3688 \times 10^{-9}$

1. Des expérimentations ont permis de constater que 20 exécutions de GRASP permettaient d'obtenir des statistiques assez précises.

TABLE 6.3: Comparaison de différents algorithmes d'optimisation sur un filtre IIR–8 (36 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	Temps (sec)	$n_{\mathcal{C}}$	n_{λ}	Énergie (J)
Glouton–a	1479	1	4120	$2,2811 \times 10^{-9}$
Glouton–ac	661	1585	1747	$1,6239 \times 10^{-9}$
CDM	5065	12535	12697	$2,3779 \times 10^{-9}$
MOGA (10)	3491	804	8170	$3,1758 \times 10^{-9}$
MOGA+ (10)	.	.	.	$1,6254 \times 10^{-9}$
MOGA (500)	20206	39522	42517	$2,1630 \times 10^{-9}$
MOGA+ (500)	.	.	.	$1,6004 \times 10^{-9}$
GRASP–a (10)	15941	1	40813	$1,5057 \times 10^{-9}$

Pour le filtre IIR 36 variables, nous observons les mêmes tendances, cependant le glouton–ac est bien meilleur que le CDM, avec un temps d'exécution très faible. Les résultats des algorithmes génétiques sont toujours de mauvaise qualité à la $10^{ième}$ génération : dans les expérimentations suivantes, nous nous sommes intéressés uniquement aux résultats à la $500^{ième}$ génération. L'algorithme proposé, GRASP–a, possède les meilleures performances. La solution de GRASP–a est environ $6,7$ % meilleure que celle de l'algorithme Génétique+ et 8 % meilleure que celle du glouton–ac. Vu la taille importante du problème, le temps d'exécution de GRASP–a est assez élevé en raison de l'exécution de nombreuses recherches locales.

TABLE 6.4: Comparaison de différents algorithmes d'optimisation sur une FFT 64 (12 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	Temps (sec)	$n_{\mathcal{C}}$	n_{λ}	Énergie (J)
Glouton–a	67	1	1731	$33,187 \times 10^{-8}$
Glouton–ac	30	470	542	$4,8112 \times 10^{-8}$
CDM	126	1623	1867	$22,146 \times 10^{-8}$
MOGA (500)	2715	40213	41412	$6,0298 \times 10^{-8}$
MOGA+ (500)	.	.	.	$5,6084 \times 10^{-8}$
GRASP–a (10)	278	165	7985	$5,2131 \times 10^{-8}$

Le résultat pour un problème de plus petite taille, une FFT à 12 variables, est présenté dans le tableau 6.4. Dans ce cas, le glouton–a et CDM se retrouvent dans une mauvaise direction et le résultat est loin d'être comparable avec les autres algorithmes. Le glouton–ac trouve la meilleure solution qui est 8 % meilleure que le résultat de GRASP–a. Dans la table 6.5 nous observons des résultats similaires mais dans ce cas, le glouton–ac est seulement $0,26$ % meilleur que GRASP–a. Ces différents résultats sur la FFT montrent que le critère $\nabla_{k/\lambda\mathcal{C}}$ est souvent un meilleur choix par apport à $\nabla_{k/\lambda}$. Le glouton–ac étant basée sur $\nabla_{k/\lambda}$, elle a sur GRASP–a l'avantage du critère de choix de direction. Dans les expérimentations avec GRASP–ac sur les mêmes problèmes, nous observerons

que GRASP–ac est meilleur que le glouton–ac.

TABLE 6.5: Comparaison de différents algorithmes d'optimisation sur une FFT 64 (20 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	Temps (sec)	$n_{\mathcal{C}}$	n_{λ}	Énergie (J)
Glouton–a	88	1	6154	$28,237 \times 10^{-8}$
Glouton–ac	60	1402	1522	$1,8990 \times 10^{-8}$
CDM	167	3963	4324	$8,6889 \times 10^{-8}$
MOGA (500)	2715	44770	44890	$2,3745 \times 10^{-8}$
MOGA+ (500)	.	.	.	$2,0782 \times 10^{-8}$
GRASP (10)	10923	53643	117267	$1,9401 \times 10^{-8}$

Dans les tableaux 6.6 et 6.7 sont présentés les résultats pour le filtre adaptatif NLMS 128 points avec 25 et 49 variables. Les algorithmes glouton–ac et GRASP–a fournissent les meilleurs solutions.

TABLE 6.6: Comparaison de différents algorithmes d'optimisation sur un NLMS 128 (25 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	Temps (sec)	$n_{\mathcal{C}}$	n_{λ}	Énergie (J)
Glouton–a	47	1	426	$2,1495 \times 10^{-8}$
Glouton–ac	60	302	452	$2,1265 \times 10^{-8}$
CDM	55	277	439	$2,1495 \times 10^{-8}$
MOGA (500)	6001	42084	42234	$2,5126 \times 10^{-8}$
MOGA+ (500)	.	.	.	$2,1265 \times 10^{-8}$
GRASP–a (10)	414	395	3630	$2,1265 \times 10^{-8}$

Afin de faciliter la comparaison pour les filtres IIR, la qualité des solutions et le temps d'exécution des algorithmes sont illustrés aux figures 6.4 et 6.5. Les résultats pour un filtre IIR 14 variables sont aussi présentés, mais peu de différences entre les algorithmes sont observables. La structure d'un filtre IIR est simple et les résultats ne sont pas très différents. Quant au temps d'exécution, les algorithmes de recherche locale, qu'ils soient déterministes ou stochastiques, nécessitent plus de temps lorsque la taille du problème augmente. Cela est dû au fait que la distance de la CLM à la solution, en général, augmente lors que le nombre de variables augmente. Les algorithmes de recherche locale utilisent donc plus de temps pour arriver à la solution. Les algorithmes génétiques, quant à eux, utilisent un nombre de calcul fixe indépendant du nombre de variables. Le temps d'exécution des algorithmes génétiques ne dépend que du nombre de générations et du temps d'évaluation de la fonction d'adaptation.

Dans les expérimentations présentées ci-dessus, les performances des algorithmes ont été analysées à travers des problèmes de tailles différentes. Cependant, pour la même application et le même nombre de variables, les performances des algorithmes d'optimisation

TABLE 6.7: Comparaison de différents algorithmes d'optimisation sur un NLMS 128 (49 variables). La première colonne contient le nom de l'algorithme avec le nombre de générations ou d'itérations dans les parenthèses. $n_{\mathcal{C}}$ et n_{λ} sont respectivement le nombre d'évaluations des fonctions de coût et de qualité (machine : PC1).

Algorithme	Temps (sec)	$n_{\mathcal{C}}$	n_{λ}	Énergie (J)
Glouton–a	245	1	2059	$2,1495 \times 10^{-8}$
Glouton–ac	353	1864	2158	$2,1110 \times 10^{-8}$
MOGA (500)	7107	46268	46562	$3,0999 \times 10^{-8}$
MOGA+ (500)	.	.	.	$2,1110 \times 10^{-8}$
GRASP–a (10)	2396	1356	20025	$2,1110 \times 10^{-8}$

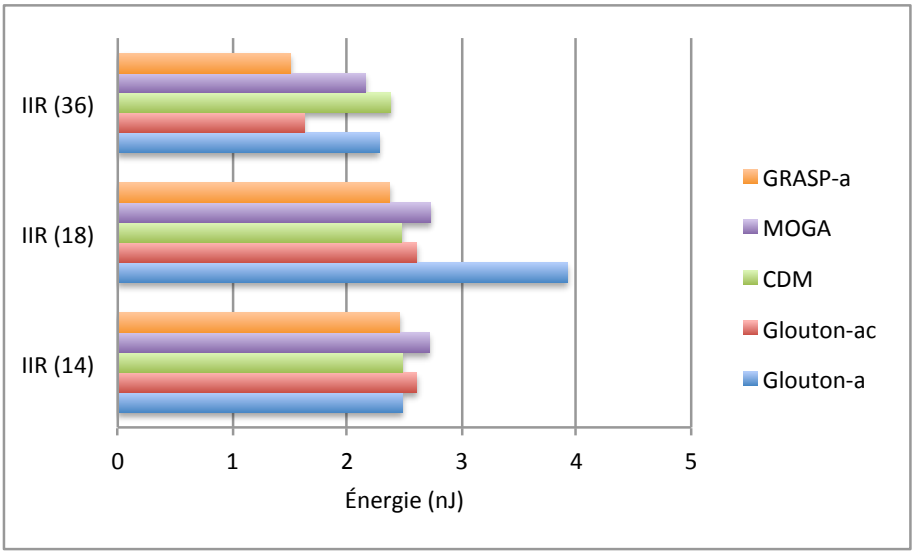


FIGURE 6.4: Énergie en 10^{-9} J obtenue pour différents algorithmes d'optimisation (glouton–a, glouton–ac, CDM, MOGA et GRASP–a) sur plusieurs configurations des filtres IIR pour une contrainte de précision de P_b fixée à $P_{b,\max} = -55$ dB (machine : PC1).

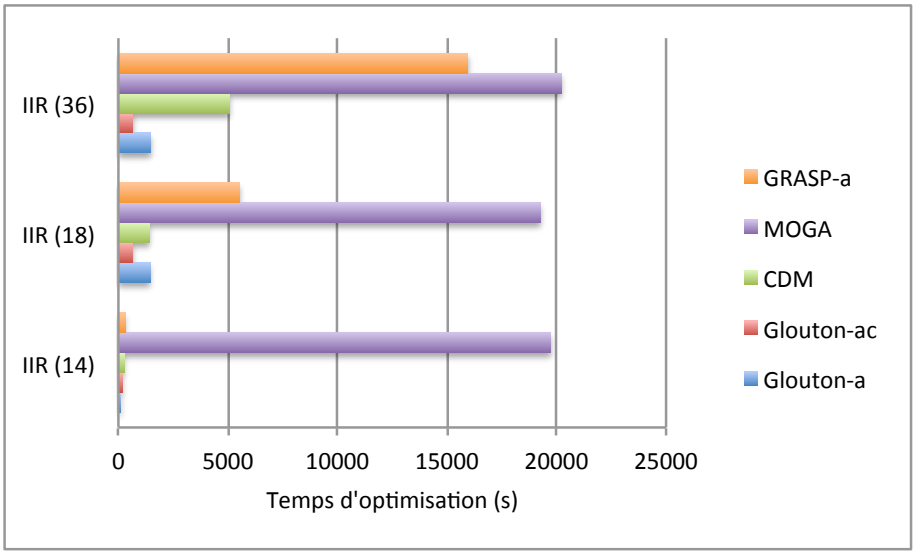


FIGURE 6.5: Temps d'exécution en seconde pour différents algorithmes d'optimisation (glouton–a, glouton–ac, CDM, MOGA et GRASP–a) sur plusieurs configurations des filtres IIR. Dans ces trois configurations, le temps d'optimisation des algorithmes génétiques reste le même. Les algorithmes de recherche locale nécessite plus de temps quand la taille du problème augmente (machine : PC1).

évoluent en fonction de la contrainte de précision. Pour un même filtre IIR, la performance de chaque algorithme varie en fonction de la puissance maximale acceptée du bruit de quantification. Dans la figure 6.6, le surcoût relatif par rapport à la meilleure solution obtenue est présenté dans le cas d'un filtre IIR pour des contraintes de précision allant de –65 dB à –50 dB. Nous observons que les performances relatives entre les différents algorithmes dépend de la contrainte de précision. Cependant, GRASP–a fournit en général la meilleure solution.

Par rapport à la recherche locale classique (glouton–a) qui utilise aussi la métrique $\nabla_{k/\lambda}$, la recherche locale stochastique (GRASP–a) est toujours meilleure. Pour les applications de taille petite (9 variables, 13 et 14 variables), la différence entre GRASP–a et le glouton est faible (2 % – 4 %). Quand le nombre de variables augmente à 36, la différence moyenne est environ de 80%. Dans la partie suivante, nous obtiendrons la même tendance entre GRASP–ac et le glouton–ac.

6.4.2 Algorithme GRASP–ac

Dans la partie précédente, nous avons comparé la performance de GRASP–a par rapport aux différents algorithmes d'optimisation. Nous avons remarqué que le glouton–ac, en utilisant la métrique $\nabla_{k/\lambda\mathcal{C}}$, est plus performante dans plusieurs cas, particulièrement pour la FFT et le NLMS. Les expérimentations dans cette partie comparent GRASP–ac et le glouton–ac qui utilisent la même métrique $\nabla_{k/\lambda\mathcal{C}}$. Ainsi, pour montrer la performance de GRASP–ac, l'algorithme s'arrête après la $5^{ième}$ itération au lieu de la $10^{ième}$ itération comme dans le GRASP–a.

La comparaison des performances du glouton–ac et du GRASP–ac pour les problèmes d'optimisation des largeurs dans le cas de la FFT est présentée à la figure 6.7. Comme nous

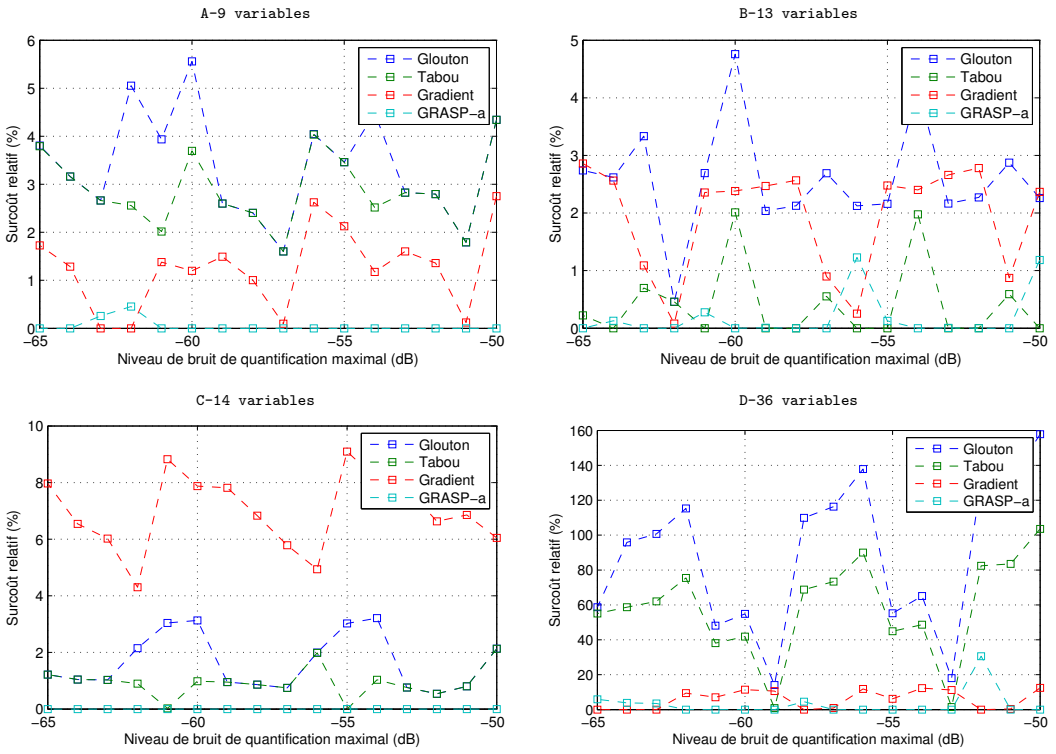


FIGURE 6.6: Comparaison des performances de différents algorithmes d'optimisation (glouton–a, tabou, glouton–ac et GRASP–a) sur un filtre IIR pour différents nombres de variables et pour différents critères de précision. Dans la plupart des cas, la meilleure solution est obtenue par l'algorithme GRASP–a (machine : PC2).

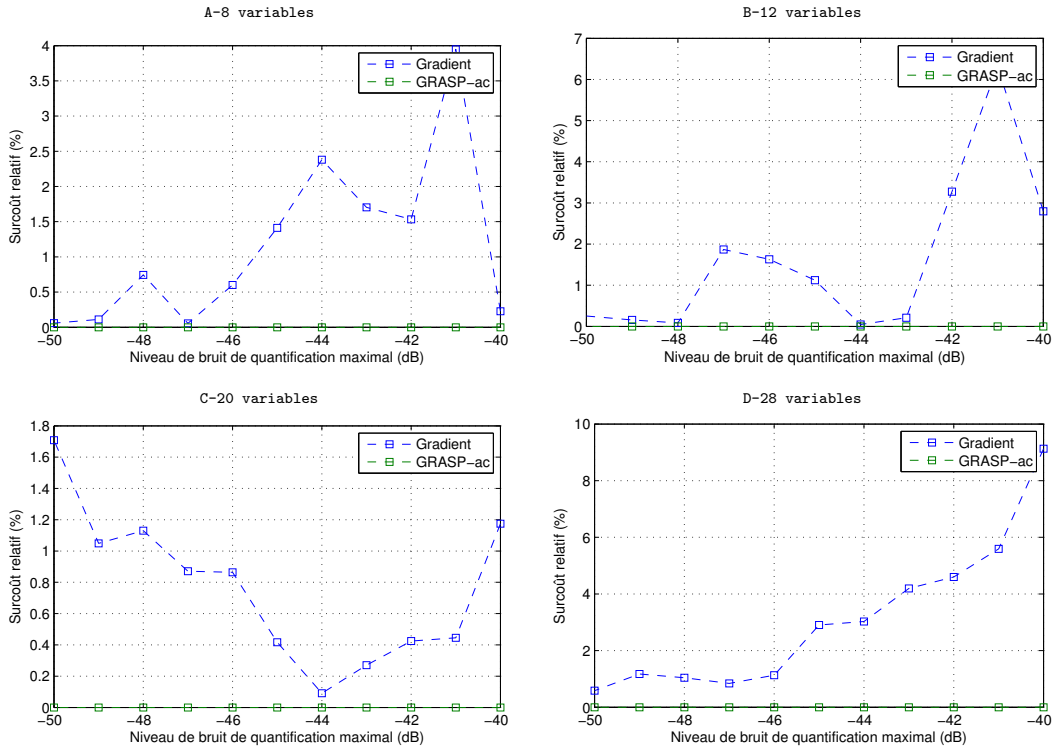


FIGURE 6.7: Comparaison des performances de deux algorithmes d'optimisation glouton–ac et GRASP–ac sur une FFT pour différents nombres de variables et pour différents critères de précision (machine : PC2).

pouvons constater, le résultat de GRASP–ac est toujours meilleur que celui du glouton–ac. Cette différence dépend de la contrainte de précision, variant entre –50 dB et –40 dB. Sur une FFT à 28 variables, GRASP–ac est jusqu'à 10% meilleur que le glouton–ac.

Les mêmes expérimentations sont réalisées sur les NLMS 128 points à 7, 10, 13, 18, 25 et 49 variables pour 11 niveaux de la puissance du bruit de quantification compris entre –45 dB et –35 dB. La différence entre GRASP–ac et le glouton–ac est plus importante que dans le cas de la FFT. Pour le NLMS à 7 variables, dans 3/11 des cas, la différence est comprise entre 15% et 20%. À 10 variables, dans 4/11 des cas, la différence est comprise entre 8% et 10%. À 49 variables, dans 6/11 des cas, la différence est comprise entre 8% et 12%.

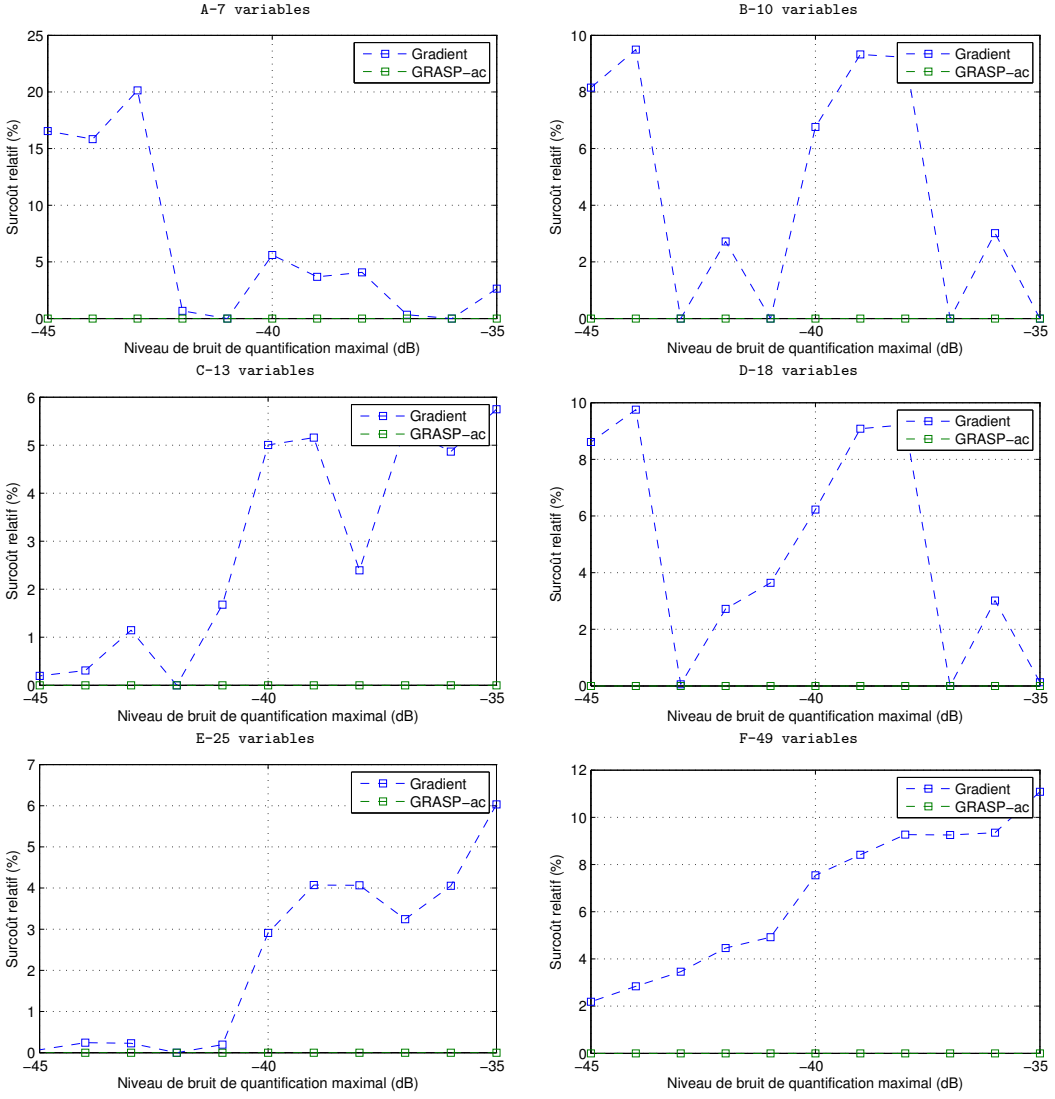


FIGURE 6.8: Comparaison des performances de deux algorithmes d'optimisation glouton–ac et GRASP–ac sur un NLMS pour différents nombres de variables et pour différents critères de précision (machine : PC2).

6.4.3 Combinaison de GRASP–a et GRASP–ac

Nous avons pu observer que chaque variante de GRASP est meilleure en termes de performance que l'algorithme déterministe correspondant. Pour obtenir la meilleure solution dans tous les cas, nous combinons ces deux variantes de GRASP en les exécutant indépendamment et en retenant le meilleur résultat. Cet algorithme est donc appelé GRASP–a/ac. Afin de mesurer l'amélioration potentielle de GRASP–a/ac par apport au glouton-a, au glouton–ac et à la combinaison de ces deux algorithmes, nous réalisons 189 tests à travers les trois applications IIR, FFT et NLMS avec différentes contraintes de précision et différents nombres de variables. La plus grande différence entre GRASP–a/ac et les autres algorithmes sur ces applications est présentée à la figure 6.9. 100 % représente la meilleure solution. Sur le filtre IIR, GRASP–a/ac peut améliorer de 158 % la solution par rapport aux gloutons. En prenant algorithme combinant les métriques de glouton et du glouton–ac, l'amélioration est 14 %. L'amélioration pour la FFT et le NLMS est de 9 % et 20 % par rapport à la combinaison des algorithmes de recherche locale déterministes.

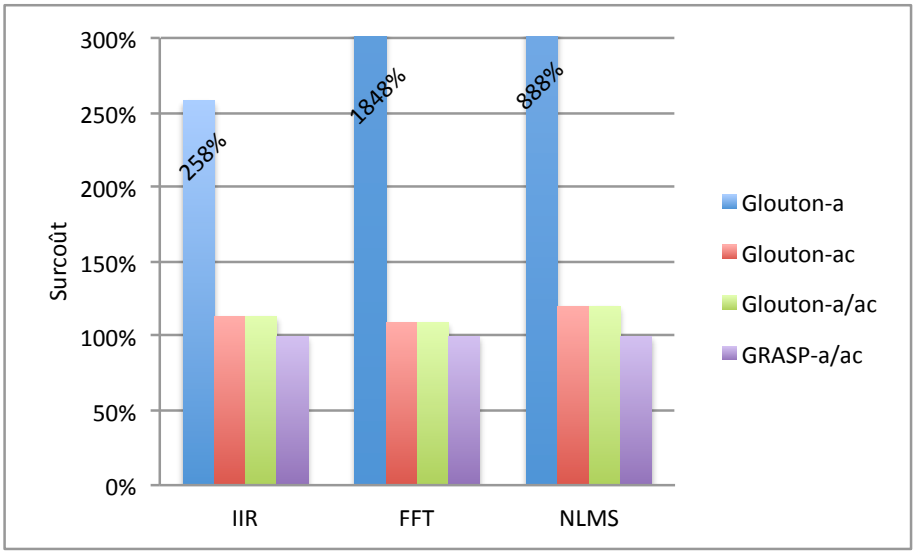


FIGURE 6.9: Performance des algorithmes d'optimisation des largeurs à travers de trois familles d'application : IIR, FFT et NLMS. Les résultats sont normalisés : 100% est la meilleure solution. La combinaison GRASP–a/ac obtient toujours la meilleure performance.

6.5 Conclusions

Étant donnée l'étendue des algorithmes de recherche locale, GRASP a montré ses performances pour l'optimisation des largeurs. Les résultats de GRASP dans tous les cas sont meilleurs que ceux des algorithmes « classiques » comme les algorithmes gloutons, CDM et des algorithmes plus modernes comme les algorithmes génétiques.

Les algorithmes de recherche locale déterministes requièrent un temps d'exécution faible pour trouver une solution. Cette solution n'est pas optimale et les algorithmes déterministes ne peuvent pas améliorer cette solution dans le cas où plus de temps est accordé.

GRASP est un algorithme stochastique de complexité moyenne permettant de trouver une meilleure solution si plus de temps est alloué à l'optimisation.

Comme toutes les heuristiques, plusieurs paramètres peuvent influencer la performance de GRASP. Nous pouvons envisager de modifier la taille de la liste RCL (par une valeur fixe non-entière, par un pourcentage ou en adaptant la taille pendant l'optimisation – le GRASP *réactif*), car une valeur fixe de la taille peut l'écarter des bonnes solutions [121]. Apparemment, la taille de la liste est dépendante du nombre d'itérations. Ainsi, un biais possible peut être introduit dans le classement des éléments dans la RCL.

Un avantage évident de GRASP est la parallélisation car toutes les itérations peuvent être exécutées en même temps. Cela permettra d'accélérer l'exécution de GRASP en fonction du nombre de processeurs disponibles.

Conclusions et perspectives

L'implantation efficace d'applications de traitement numérique du signal dans un système embarqué requiert l'utilisation de l'arithmétique virgule fixe. Le codage manuel des données en virgule fixe est une tâche fastidieuse, longue et source d'erreurs. En outre, la réduction du temps de mise sur le marché nécessite l'utilisation d'outils de haut-niveau. Ces outils intègrent des algorithmes d'optimisation permettant de trouver la combinaison des largeurs optimisée, laquelle a pour objectif de minimiser la fonction de coût et de satisfaire les contraintes de précision.

De plus, dans un système de communications sans fil, le canal de transmission est souvent fluctuant. Ceci donne une opportunité d'adapter le format des données en fonction de la qualité du canal afin de minimiser la consommation d'énergie.

Ce travail de recherche s'articule autour de deux axes principaux. Dans un premier temps, le principe d'adaptation dynamique de la précision a été introduit. La largeur des données est reconfigurée et optimisée en fonction du rapport signal à bruit (RSB) à l'entrée. Nous avons montré l'intérêt de notre approche et défini le support d'exécution de l'architecture. Des études sont réalisées pour déterminer analytiquement la spécification virgule fixe en fonction du RSB pour les systèmes QPSK et DS-CDMA. Avec le même critère de précision, par rapport à l'approche classique, nous pouvons potentiellement économiser jusqu'à 40 % de la consommation d'énergie dans le récepteur *Rake* et jusqu'à 25 % dans le module de recherche de trajets d'un récepteur WCDMA.

Dans un second temps, une étude sur les algorithmes d'optimisation des largeurs est présentée. Ces algorithmes essayent, à partir d'un critère de précision, de trouver la meilleure combinaison des largeurs conduisant à un coût minimal. Nous avons apporté des améliorations aux algorithmes génétiques pour les systèmes ayant un nombre important d'opérations. Cette contribution permet de trouver la meilleure solution avec un nombre de générations plus faible par rapport aux algorithmes classiques.

Pour des systèmes de taille moyenne, une recherche locale stochastique basée sur GRASP est proposée. Ces algorithmes sont meilleurs que les algorithmes classiques en termes de performance et de compromis entre performance et complexité de calcul. Une variation de l'algorithme glouton min+1 et la recherche tabou ont été également proposées. Cet algorithme GRASP peut trouver une solution plus satisfaisante en gardant la simplicité des algorithmes de recherche locale.

Perspectives

Les résultats sur le pourcentage d'énergie économisée sont basés sur l'hypothèse que le RSB à l'entrée du récepteur est distribué sur un intervalle de 20 dB. En réalité, cet intervalle dépend de plusieurs paramètres, comme la qualité de service (QoS) et le débit souhaité. Un scénario réel est également simulé, mais il reste assez simple. Il serait intéressant d'avoir *des scénarios plus complexes* en tenant compte des spécifications du WCDMA. De plus, il est nécessaire d'évaluer les gains sur une plate-forme réelle.

La seconde perspective concerne une alternative à notre approche. Au lieu de fixer le taux d'erreur binaire (TEB) en virgule fixe en fonction du TEB en virgule flottante, nous pouvons nous intéresser à un TEB unique. Cette valeur, définie sur un intervalle de RSB plus étroit, permet de garantir la qualité nécessaire de la communication. Cette approche maintient le bruit total correspondant au TEB demandé. Le rapport signal sur bruit de quantification (RSBQ) dépend tout de même du RSB mais sous une relation différente que celle utilisée dans notre première approche.

La troisième perspective concerne des études pour déterminer le surcoût de la gestion dynamique de la précision. Jusqu'à maintenant, nous supposons que ce surcoût est négligeable et celui-ci n'a pas été pris en compte. En effet, le processus d'optimisation est exécuté hors ligne. Cependant, la surveillance de la qualité de transmission et la gestion de la reconfiguration possède un certain coût.

Une autre perspective de ce travail de recherche se situe au niveau des algorithmes d'optimisation. Nous avons proposé l'élitisme et la recherche locale pour l'algorithme génétique. Nous pourrions utiliser *le classement rapide* des individus proposé dans NSGA–II afin de diminuer le temps de calcul. Sans oublier que d'autres techniques présentées dans 5.3.2 peuvent être utilisées.

Une dernière perspective concerne la méta-heuristique GRASP. Les expérimentations montrent que le temps d'optimisation avec GRASP augmente rapidement lorsque la taille du problème augmente et cet algorithme est donc inapproprié aux problèmes de taille très grande. Des techniques permettant d'améliorer le résultat peuvent être envisagées, y compris pour choisir la bonne valeur de *la taille de liste de candidats* (RCL) afin que la recherche soit plus directionnelle sans perdre de la diversité. Une autre technique concerne l'adaptation du nombre d'itérations en fonction de la taille du problème.

Appendices

Annexe A

Familles des algorithmes évolutionnistes

Depuis la première application de l'évolution dans les problèmes d'optimisation dans les années 1960, trois axes de développement des algorithmes évolutionnistes sont les stratégies d'évolution, la programmation évolutionniste et les algorithmes génétiques. Il existe d'autres branches développées récemment comme la programmation génétique ou l'évolution différentielle. La programmation génétique [86] a pour but l'optimisation des programmes exécutables représentés sous formes d'un arbre. Cette représentation est différente de ceux examinés par sa non-linéarité et par sa longueur variable. Ainsi, pour les problème d'optimisation dans l'espace continue, l'évolution différentielle [140] a montré ses avantages devant les algorithmes évolutionnistes classiques. Cependant, ces algorithmes n'attirent pas les chercheurs aussi que les trois principales familles.

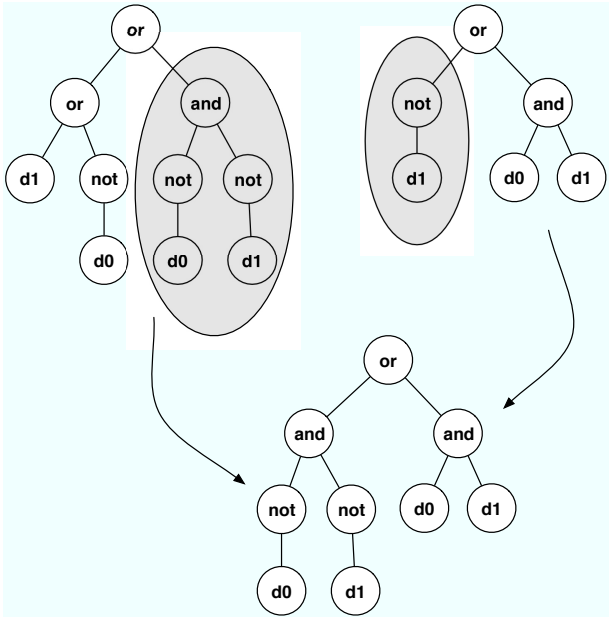


FIGURE A.1: La recombinaison de programme exécutable présenté en arbre. Source : [8]

A.1 Stratégies d'évolution

La méthode, denomé SE (ou ES, pour *Evolution Strategies* en anglais), est le premier algorithme évolutionniste et a été proposé par Ingo Rencherberg, en 1965[125]. Cet algorithme a été initialement utilisé dans la conception de profils aérodynamiques et dans les problèmes d'optimisation continus. La première version de l'algorithme ne manipule qu'un seul individu et est donc noté $(1+1)$–ES. À chaque génération, un enfant est reproduit par mutation à partir du parent et l'un ou l'autre sera sélectionné pour la génération suivante.

Rencherberg a ensuite proposé l'algorithme $(\mu + 1)$–ES qui signifie que μ parents génèrent un enfant pour chaque génération. Cette stratégie, n'a pas été largement utilisé, mais a donné les idées de base pour une transition à $(\mu + \lambda)$–ES et (μ, λ)–ES [135] ((μ, λ) signifie que l'étape de sélection s'applique uniquement sur λ enfants). De plus, ces derniers algorithmes utilisent également la recombinaison. Ces améliorations ont permis aux SE de résoudre non seulement les problèmes d'optimisation continus mais aussi ceux d'optimisation combinatoire. Cette approche reste encore compétitive aujourd'hui. Bäck a présenté un panorama de la théorie des stratégies d'évolution dans [9].

Représentation des paramètres Dans une stratégie d'évolution, chaque individu $\boldsymbol{a} \in \boldsymbol{I}$ ($\boldsymbol{I}$ désigné l'espace des individus) est composé d'une variable d'objectif $\boldsymbol{x}$ et des paramètres de stratégie. La variable d'objectif est représentée par un point $\boldsymbol{x}$ dans l'espace $\mathcal{R}^n$. La fonction d'adaptation est identique à la fonction d'objectif. Les paramètres de stratégie sont spécifiés par n variances $c_{ii} = \sigma_i^2$ et $n \cdot (n-1)/2$ covariances c_{ij} d'une loi normale multidimensionnelle admettant la densité de probabilité suivante :

$$p(\boldsymbol{z}) = \sqrt{\frac{\det \boldsymbol{A}}{(2\pi)^n}} \exp\left(-\frac{1}{2}{}^{\mathrm{t}}\boldsymbol{z}\boldsymbol{A}\boldsymbol{z}\right) \tag{A.1}$$

où $\boldsymbol{A}^{-1} = (c_{ij})$ est la matrice de variance – covariance et $\boldsymbol{z}$ le vecteur aléatoire de dimension n. Un individu $\boldsymbol{a} \in \boldsymbol{I}$ peut donc prendre jusqu'à $w = n \cdot (n+1)/2$ paramètres de stratégie : $\boldsymbol{I} = \mathcal{R}^{n+w}$. Cependant, dans la plupart de cas, seul les variances sont prises en compte ainsi $\boldsymbol{I} = \mathcal{R}^{2n}$. Parfois une variance commune est utilisée pour tous les composantes de la variable d'objectif, i.e. $\boldsymbol{I} = \mathcal{R}^{n+1}$.

Mutation La mutation est l'opérateur le plus important dans une stratégie d'évolution. Cet opérateur modifie les paramètres statistiques d'un individu $\boldsymbol{a}$ ainsi que sa variable d'objectif $\boldsymbol{x}$. Afin d'assurer que la matrice variance – covariance soit définie positive, les paramètres de stratégie sont représentés sous forme des angles de rotation :

$$\alpha_{ij} = \frac{1}{2} \arctan\left(\frac{2c_{ij}}{\sigma_i^2 - \sigma_j^2}\right) \tag{A.2}$$

Un opérateur de mutation $m'_{(\tau,\tau',\beta)} : \boldsymbol{I} \to \boldsymbol{I}$ est défini par :

$$m'_{(\tau,\tau',\beta)}(\boldsymbol{x}, \boldsymbol{\sigma}, \boldsymbol{\alpha}) = (\boldsymbol{x}', \boldsymbol{\sigma}', \boldsymbol{\alpha}') \tag{A.3}$$

avec

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \\ \alpha_j' &= \alpha_j + \beta \cdot N_j(0,1) \\ \boldsymbol{x}' &= \boldsymbol{x} + \vec{N}(\mathbf{0}, \boldsymbol{\sigma}', \boldsymbol{\alpha}') \end{aligned} \tag{A.4}$$

Les coefficients τ, τ', β permettent de modifier le comportement de l'opérateur. Le facteur τ modifie globalement la mutation alors que τ' modifie la valeur moyenne de chaque σ_i. Schwefel propose que la valeur β soit égale à 5° $(0,0873$ en radian).

Recombinaison L'opérateur de recombinaison n'est utilisé que dans les versions améliorées des stratégies d'évolution. Soient $\boldsymbol{x}_{S,i}$ et $\boldsymbol{x}_{T,i}$ les variables d'objectif de deux parents choisis uniformément dans la génération actuelle. La variable d'objectif de l'enfant est calculée à partir de l'expression suivante :

$$\boldsymbol{x}_i' = \boldsymbol{x}_{S,i} + \chi \cdot (\boldsymbol{x}_{T,i} - \boldsymbol{x}_{S,i}) \tag{A.5}$$

avec χ variable aléatoire uniformément distribuée sur $[0,1]$. Historiquement, la valeur de χ était fixée χ à $1/2$.

Sélection La sélection est un opérateur déterministe (sélection par troncature). Ainsi, μ meilleurs individus sont sélectionnés à partir de λ enfants ou de $\mu + \lambda$ parents et enfants. La sélection $(\mu + \lambda)$ garantit une évolution monotone alors que la sélection (μ, λ) permet de respecter les paramètres de stratégie et de favoriser l'auto-adaptation. Aujourd'hui, la sélection (μ, λ) est conseillé et Schwefel a suggéré que la valeur optimale de μ/λ soit environ égale à $1/7$[136].

A.2 Programmation évolutionniste

La programmation évolutionniste (EP, pour *Evolutionary Programming* en anglais) a été proposée par Fogel. Le nom de l'algorithme vient du fait qu'il s'agit des programmes d'ordinateur exécutables. L'évolution de la population est l'évolution de l'ensemble des programmes permettant de résoudre un problème. Le codage, ou alors la représentation de programme, est approprié au problème à résoudre. La programmation évolutionniste, comme les stratégies d'évolution, s'appuie sur le phénotype, donc la mutation.

Des premiers travaux se concentrent sur les automates d'états finis (FSM). Un automate d'états finis est une séquence d'instructions à exécuter, chacune dépend de l'entrée et de l'état actuel. Formellement, un automate d'états finis est défini par cinq paramètres

$$M = (Q, \tau, \rho, s, o) \tag{A.6}$$

ou Q est l'ensemble fini d'états, τ, l'ensemble fini d'entrées, ρ, l'ensemble fini de sorties, $s : Q \times \tau \mapsto Q$ la fonction de changement d'état et $o : Q \times \tau \mapsto s$, la fonction de sortie. L'adaptation d'un automate (individu) est calculée par la capacité de prédiction. À chaque automate est donné une séquence de symboles déjà observée, la sortie de l'automate est mesuré par apport au résultat à priori connu. Typiquement, l'évolution de génération à

génération est sous le forme $(\mu + \lambda)$–EP avec $\mu = \lambda$. Dans ce processus, chaque configuration de la population courante est copiée dans la nouvelle génération. Puis, un individu enfant est généré par une erreur stochastique (la mutation) à partir d'un individu parent. L'ensemble 2λ individus est passé à l'étape de sélection pour garder les λ meilleurs individus. Ces individus survivants génèrent la génération suivante. Il faut noter que, comme dans les stratégies d'évolution, différents critères de sélection peuvent être envisagés. Les formes de sélection sont détaillées dans la section suivante.

Dans un automate à états finis, la mutation est réalisée soit en changeant le symbole d'une sortie, l'état initial, ou un état de transition, ou soit en ajoutant ou en supprimant un état. Afin éviter les automates « null », la suppression d'un état et le changement de l'état initial ne sont permis que si le parent a plus d'un état.

Depuis la fin des années 1980, la programmation évolutionniste est adapté pour résoudre les problèmes d'optimisation combinatoire [44]. Alors qu'il nécessite un codage particulier pour chaque problème, le résultat est trouvé rapidement en restreignant l'espace de recherche sur une partie extrêment petite.

A.3 Algorithmes génétiques

Dans la famille des algorithmes évolutionnistes, les algorithmes génétiques (AG) sont les plus populaires, particulièrement après le livre révolutionnaire de Holland [72] en 1975. La différence principale entre un algorithme génétique et les autres approches génétiques est la distinction du génotype et du phénotype. Cette propriété implique que dans un algorithme génétique, le meilleur individu est composé de différents blocs, chaque bloc étant le meilleur. Le choix du codage du génotype, généralement codé de façon binaire, est critique dans cette classe d'algorithme : un bon choix est un choix dans lequel le génotype est bien relié au phénotype.

Les algorithmes génétiques utilisent toutes les étapes d'un algorithme évolutionniste. Le croisement est l'opérateur principal dans la recombinaison.

Certains algorithmes évolutionnistes utilisant d'autres représentations et opérateurs sont souvent appelés algorithmes génétiques, bien que les spécialistes évitent cet abus de langage.

Annexe B

Bibliothèque des opérateurs utilisée pour les simulations

B.1 Registres

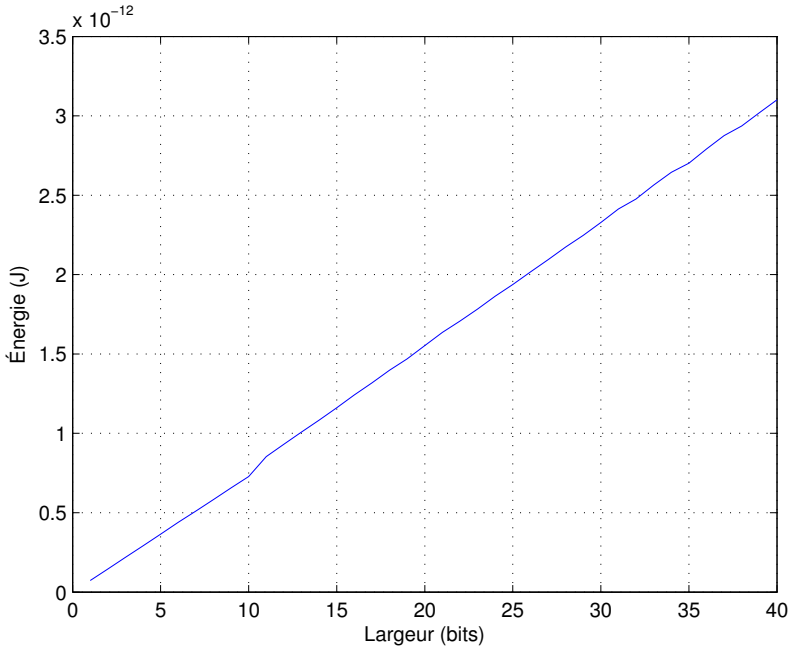


FIGURE B.1: Consommation d'énergie des registres dans notre bibliothèque.

B.2 Additionneurs

B.3 Multiplieurs

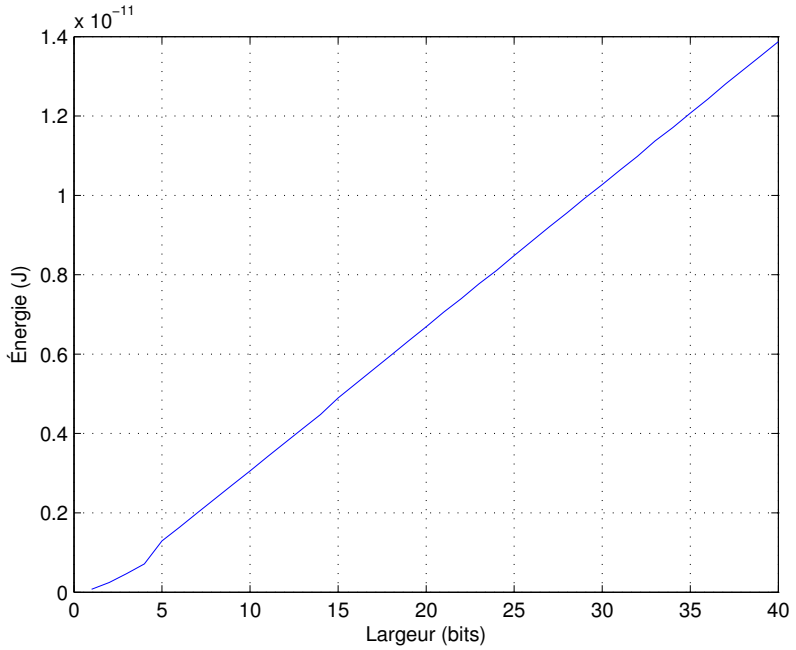


FIGURE B.2: Consommation d'énergie des additionneurs dans notre bibliothèque.

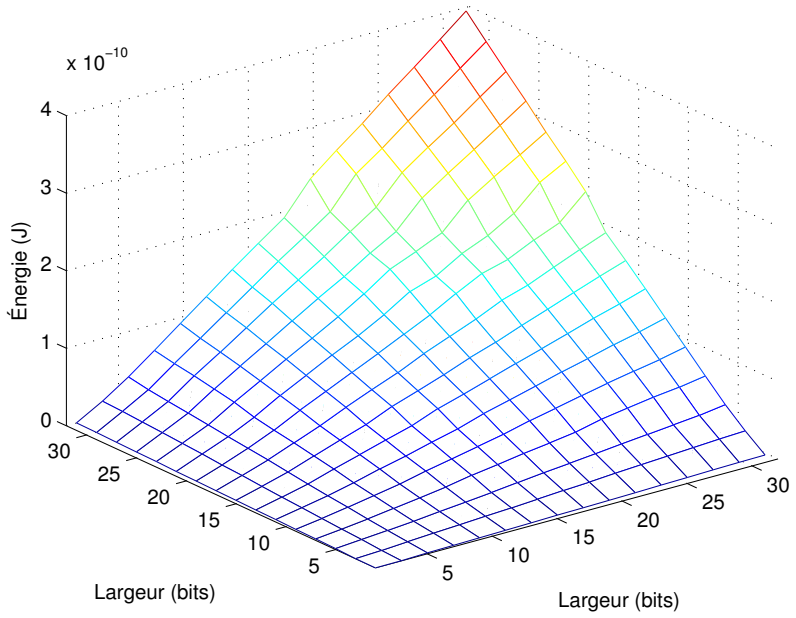


FIGURE B.3: Consommation d'énergie des multiplieurs dans notre bibliothèque.

Bibliographie

[1] 3GPP, "TS 25.104 V8.3.0 : Base Station radio transmission and reception (FDD)," 2008.

[2] A. Ahmadi and M. Zwolinski, "Word-Length Oriented Multiobjective Optimization of Area and Power Consumption in DSP Algorithm Implementation," in *Proc. International Conference on Microelectronics (MIEL)*, Belgrade, 2006, pp. 614–617.

[3] Analog Device, *TigerSHARC Hardware Specification*, Analog Device, December 1999.

[4] T. Arslan and D. Horrocks, "A genetic algorithm for the design of finite word length arbitrary response cascaded IIR digital filters," in *Proc. First International Conference on Genetic Algorithms in Engineering Systems : Innovations and Applications (GALESIA'95)*, 1995, pp. 276–281.

[5] H. Aytug and G. Koehler, "New stopping criterion for genetic algorithms," *European Journal of Operational Research*, vol. 126, no. 3, pp. 662–674, 2000.

[6] T. Bäck, *Evolutionary algorithms in theory and practice : evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, USA, 1996.

[7] T. Bäck, D. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation.* Taylor & Francis, 1997.

[8] T. Bäck, D. Fogel, and Z. Michalewicz, *Evolutionary Computation 1 : Basic Algorithms and Operators.* Taylor & Francis, May 2000, vol. 1.

[9] T. Bäck and H. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.

[10] N. Benvenuto, M. Marchesi, G. Orlandi, F. Piazza, and A. Uncini, "Finite word-length digital filter design using an annealing algorithm," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1989, pp. 861–864.

[11] M. Bhardwaj, R. Min, and A. Chandrakasan, "Power-aware systems," in *Proc. Asilomar Conference on Signals, Systems and Computers (ACSSC)*, Dec. 2000, pp. 1695–1701.

[12] T. Blickle and L. Thiele, "A mathematical analysis of tournament selection," in *Proc. International Conference on Genetic Algorithms (ICGA)*, 1995, pp. 9–16.

[13] G. Caffarena, G. Constantinides, P. Cheung, C. Carreras, and O. Nieto-Taladriz, "Optimal combined word-length allocation and architectural synthesis of digital si-

gnal processing circuits," *IEEE Transactions on Circuits and Systems—Part II : Express Briefs*, vol. 53, no. 5, pp. 339–343, 2006.

[14] M. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, 2002, pp. 612–615.

[15] M. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 5, 2001, pp. 53–56.

[16] E. Casseau and B. Le Gal, "High-level synthesis for the design of fpga-based signal processing systems," jul. 2009, pp. 25 –32.

[17] F. Catthoor, H. De Man, and J. Vandewalle, "Simulated-annealing-based optimization of coefficient and data word-lengths in digital filters," *International Journal of Circuit Theory and Applications*, vol. 16, no. 4, 1988.

[18] S. Chen and B. Luk, "Adaptive simulated annealing for optimization in signal processing applications," *EURASIP Signal Processing Journal*, vol. 79, no. 1, pp. 117–128, 1999.

[19] S. Chen, J. Wu, R. Istepanian, and J. Chu, "Optimizing stability bounds of finite-precision PID controller structures," *IEEE Transactions on Automatic Control*, vol. 44, no. 11, pp. 2149–2153, 2002.

[20] H. Choi and W. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *Proc. IEEE Workshop on VLSI Signal Processing (VLSI–SP)*, San Diego, CA, Oct. 1994, pp. 198–207.

[21] C. Coello, S. de Computación, and C. Zacatenco, "Twenty years of evolutionary multi-objective optimization : A historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, 2006.

[22] G. Constantinides, "Perturbation analysis for word-length optimization," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2003, pp. 81–90.

[23] G. Constantinides, P. Cheung, and W. Luk, "The multiple wordlength paradigm," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Rohnert Park, CA, 2001, pp. 51–60.

[24] G. Constantinides, P. Cheung, and W. Luk, "Optimum wordlength allocation," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2002, pp. 219–228.

[25] G. Constantinides, P. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 10, pp. 1432–1442, 2003.

[26] G. Constantinides, P. Cheung, and W. Luk, *Synthesis and optimization of DSP algorithms*. Kluwer Academic Publishers, 2004.

[27] G. Constantinides, P. Cheung, and W. Luk, "Optimum and heuristic synthesis of multiple word-length architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 39–57, 2005.

[28] G. Constantinides and G. Woeginger, "The complexity of multiple wordlength assignment," *Applied mathematics letters*, vol. 15, no. 2, pp. 137–140, 2002.

[29] T. Crainic, B. Le Cun, and C. Roucairol, *Parallel Combinatorial Optimization*. Wiley-Interscience, 2006, ch. Parallel Branch-and-Bound Algorithms.

[30] L. Davis, Ed., *Handbook of genetic algorithms*, 1st ed. Van Nostrand Reinhold Company, Jan. 1991.

[31] F. de Dinechin, H. D. Nguyen, and B. Pasca, "Pipelined FPGA adders," in *Proc. Field Programmable Logic and Applications*. Milano, Italy : IEEE, Aug. 2010.

[32] M. de la Maza and B. Tidor, "An analysis of selection procedures with particular attention paid to proportional and Boltzmann selection," in *Proc. International Conference on Genetic Algorithms (ICGA)*. Morgan Kaufmann Publishers Inc., 1993, pp. 124–131.

[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm : NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[34] H. Elders-Boll, "Simplified interference-based threshold rule for delay selection in DS-CDMA systems," in *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, vol. 1, Sep. 2000, pp. 77–81.

[35] ETSI, "Selection procedures for the choice of radio transmission technologies of the UMTS," 1998.

[36] B. Evans, "Modem Design, Implementation, and Testing Using NI's LabVIEW," in *National Instrument Academic Day*, Beirut, Lebanon, June 2005.

[37] J. Eyre and J. Bier, "The Evolution of DSP Processors," *IEEE Signal Processing Magazine*, vol. 17, no. 2, pp. 44–51, March 2000.

[38] J. Eyre and J. Bier, "The evolution of DSP processors," *IEEE Signal Processing Magazine*, vol. 17, no. 2, pp. 43–51, March 2000.

[39] J. Eyre and J. Bier, "DSPs court the consumer," *IEEE Spectrum*, vol. 36, no. 3, pp. 47–53, 1999.

[40] C. Fang, R. Rutenbar, and T. Chen, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," in *Proc. IEEE International Conference on Computer-Aided Design (ICCAD)*. Published by the IEEE Computer Society, 2003, pp. 275–282.

[41] C. Fang, R. Rutenbar, and P. Markus, "Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling," in *Proc. Design Automation Conference (DAC)*. ACM, 2003.

[42] T. A. Feo and M. G. Resende, "A probabilistic heuristic for a computationally difficult set covering problem* 1," *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, 1989.

[43] P. D. Fiore and L. Lee, "Closed-form and real-time wordlength adaptation," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, 1999, pp. 1897–1900.

[44] D. Fogel, "An evolutionary approach to the traveling salesman problem," *Biological Cybernetics*, vol. 60, no. 2, pp. 139–144, 1988.

[45] C. Fonseca, P. Fleming *et al.*, "Genetic algorithms for multiobjective optimization : Formulation, discussion and generalization," in *ICGA*, 1993, pp. 416–423.

[46] J. Fridman, "Sub-Word Parallelism in Digital Signal Processing," *IEEE Signal Processing Magazine*, vol. 17, no. 2, pp. 27–35, March 2000.

[47] B. Gendron and T. G. Crainic, "Parallel branch-and-bound algorithms : survey and synthesis," *Operations research*, vol. 42, no. 6, pp. 1042–1066, 1994.

[48] P. Gentili, F. Piazza, A. Uncini, P. Gentili, F. Piazza, and A. Uncini, "Efficient Genetic Algorithm Design for Power-of-Two FIR Filter," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Citeseer, 1995, pp. 1268–1271.

[49] A. Gillies, "Machine learning procedures for generating image domain feature detectors." Ph.D. dissertation, University of Michigan, 1985.

[50] F. Glover, "Tabu Search – Part I," *INFORMS Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[51] F. Glover, "Tabu Search – Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[52] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[53] G. Goossens and al., "Embedded Software in Real-Time Signal Processing Systems : Design Technologies," *Proceedings of the IEEE*, vol. 85, no. 3, pp. 436–453, March 1997.

[54] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[55] T. Grötker, E. Multhaup, and O.Mauss, "Evaluation of HW/SW Tradeoffs Using Behavioral Synthesis," in *7th International Conference on Signal Processing Applications and Technology (ICSPAT 96)*, Boston, US, October 1996, pp. 781–785.

[56] E. J. Gumbel, *Statistics of Extremes*. Colombia University Press, 1958.

[57] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for CDMA demodulator," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, Sydney, May 2001, pp. 290–293.

[58] K. Han and B. Evans, "Wordlength optimization with complexity-and-distortion measure and its application to broadband wireless demodulator design," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5, 2004, pp. 37–40.

[59] K. Han, "Automating transformations from floating-point to fixed-point for implementing digital signal processing algorithms," Ph.D. dissertation, University of Texas at Austin, 2006.

[60] K. Han and B. L. Evans. (2006, May) Floating–Point to Fixed–Point Transformation Toolbox.

[61] K. Han and B. L. Evans, "Optimum wordlength search using sensitivity information," *EURASIP Journal on Applied Signal Processing*, pp. 1–14, 2006.

[62] K. Han, B. L. Evans, and E. E. Swartzl, "Low-power multipliers with data wordlength reduction," in *Proc. Asilomar Conference on Signals, Systems and Computers (ACSSC)*, Oct./Nov. 2005, pp. 1615–1619.

[63] J. Hao, P. Galinier, and M. Habib, "Metaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes," *Revue d'Intelligence Artificielle*, vol. 2, no. 13, pp. 283–324, 1999.

[64] G. Harik, E. Cantú-Paz, D. Goldberg, and B. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," *Evolutionary Computation*, vol. 7, no. 3, pp. 231–253, 1999.

[65] J. Hart and A. Shogan, "Semi-greedy heuristics : An empirical study," *Operations Research Letters*, vol. 6, pp. 107–114, 1987.

[66] M. Haseyama and D. Matsuura, "A filter coefficient quantization method with genetic algorithm, including simulated annealing," *IEEE Signal Processing Letters*, vol. 13, no. 4, pp. 189–192, 2006.

[67] S. D. Haynes, A. B. Ferrari, and P. Y. K. Cheung, "Flexible reconfigurable multiplier blocks suitable for enhancing the architecture of fpgas," in *Proceedings of Custom Integrated Circuit Conference*, 1999, pp. 16–19.

[68] N. Herve, D. Menard, and O. Sentieys, "Data wordlength optimization for FPGA synthesis," *IEEE Workshop on Signal Processing Systems (SiPS'05)*, pp. 623–628, 2005.

[69] N. Hervé, "Contributions à la synthèse d'architecture virgule fixe à largeurs multiples," Ph.D. dissertation, University of Rennes 1, ENSSAT, IRISA„ 2007.

[70] N. Hervé, "Contributions à la synthèse d'architecture virgule fixe à largeurs multiples," Ph.D. dissertation, Université de Rennes 1, 2007.

[71] R. Hinterding, H. Gielewski, and T. Peachey, "The Nature of Mutation in Genetic Algorithms," in *Proc. International Conference on Genetic Algorithms (ICGA)*. Morgan Kaufmann Publishers Inc., 1995, pp. 65–72.

[72] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[73] H. Holma and A. Toskala, *WCDMA for UMTS : Radio Access for Third Generation Mobile Communications*. Wiley, 2004.

[74] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto Genetic Algorithm for Multiobjective Optimization," in *Proc. IEEE Conference on Evolutionary Computation (ICEC'94)*, 1994, pp. 82–87.

[75] IEEE-SA Standards Board, *IEEE Standard for Floating–Point Arithmetic*, IEEE Computer Society, Jun. 2008.

[76] L. Ingber and B. Rosen, "Genetic algorithms and very fast simulated reannealing : A comparison," *Mathematical and Computer Modelling*, vol. 16, pp. 87–87, 1992.

[77] R. Kacelenga, P. Graumann, and L. Turner, "Design of digital filters using simulated annealing," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 1990, pp. 642–645.

[78] R. Kearfott, "Interval computations : Introduction, uses, and resources," *Euromath Bulletin*, vol. 2, no. 1, pp. 95–112, 1996.

[79] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE : A fixed-point design and simulation environment," in *Proc. IEEE/ACM conference on Design, Automation and Test in Europe (DATE)*. IEEE Computer Society Washington, DC, USA, 1998, pp. 429–435.

[80] S. Kim, K. Kum, and W. Sung, "Fixed-point optimization utility for C and C++ based digital signal processing programs," *IEEE Transactions on Circuits and Systems—Part II : Analog and Digital Signal Processing*, vol. 45, no. 11, pp. 1455–1464, 1998.

[81] S. Kim and W. Sung, "A Floating-point to Fixed-point Assembly program Translator for the TMS 320C25," *IEEE Transactions on Circuits and Systems*, vol. 41, no. 11, pp. 730–739, November 1994.

[82] R. Kinnison, *Applied Extreme Value Statistics.* Macmillan Publishing Company, 1985.

[83] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[84] J. Knowles and D. Corne, "The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation," in *Proc. 1999 Congress on Evolutionary Computation (CEC'99).* Piscataway, NJ : IEEE Press, Jul. 1999, pp. 98–105.

[85] A. Kountouris, "A randomized algorithm for controlling the round-off error accumulation in recursive digital frequency synthesis (DFS)," *Digital Signal Processing*, vol. 19, no. 4, pp. 534–544, 2009.

[86] J. Koza and R. Poli, "Genetic programming," *Search Methodologies*, pp. 127–164, 1992.

[87] K. Kum and W. Sung, "Word-length optimization for high-level synthesis of digital signal processing systems," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS)*, 1998, pp. 569–578.

[88] K. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 8, pp. 921–930, 2001.

[89] P. Larranaga and J. Lozano, *Estimation of distribution algorithms : A new tool for evolutionary computation.* Springer Netherlands, 2002.

[90] D. Lee, A. Gaffar, R. Cheung, O. Mencer, W. Luk, and G. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[91] A. Liefooghe, L. Jourdan, T. Legrand, J. Humeau, and E. Talbi, *Advances in Multi-Objective Nature Inspired Computing.* Springer, 2009, ch. ParadisEO-MOEO : A Software Framework for Evolutionary Multi-Objective Optimization, pp. 87–117.

[92] R. Lin, "Reconfigurable Parallel Inner Product Processor Architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, pp. 261–272, Apr. 2001.

[93] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

[94] T. Lukasiak., "Extended-precision fixed-point arithmetic on the blackfinæ processor platform," Analog Device, Tech. Rep., 2003.

[95] A. Mallik, D. Sinha, P. Banerjee, and H. Zhou, "Low-Power Optimization by Smart Bit-Width Allocation in a SystemC-Based ASIC Design Environment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 3, p. 447, 2007.

[96] D. Menard, D. Chillet, and O. Sentieys, "Floating-to-Fixed-Point Conversion for Digital Signal Processors," *EURASIP Journal on Applied Signal Processing*, vol. 14, 2006.

[97] D. Menard, R. Rocher, and O. Sentieys, "Analytical Fixed-Point Accuracy Evaluation in Linear Time-Invariant Systems," *IEEE Transactions on Circuits and Systems—Part I : Regular Papers*, vol. 55, no. 10, pp. 3197–3208, Nov. 2008.

[98] D. Menard, R. Rocher, O. Sentieys, and O. Serizel, "Accuracy Constraint Determination in Fixed-Point System Design," *EURASIP Journal on Embedded Systems*, vol. 2008, no. Article ID 242584, 2008.

[99] D. Ménard, "Méthodologie de compilation d'algorithmes de traitement du signal pour les processeurs en virgule fixe sous contrainte de précision," Ph.D. dissertation, Université de Rennes 1, 2002.

[100] D. Menard, E. Casseau, S. Khan, O. Sentieys, S. Chevobbe, S. Guyetant, and R. David, "Reconfigurable operator based multimedia embedded processor," *Reconfigurable Computing : Architectures, Tools, and Applications (Proc. ARC'09)*, vol. 5453, pp. 39–49, Mar. 2009.

[101] D. Menard, H.-N. Nguyen, F. Charot, S. Guyetant, J. Guillot, E. Raffin, and E. Casseau, "Exploiting reconfigurable SWP operators for multimedia applications," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 1717–1720.

[102] D. Menard, O. Sentieys, N. Herve, and H.-N. Nguyen, "High-level synthesis under fixed-point accuracy constraint," *Journal of Electrical and Computer Engineering*, pp. 1–26, Jan. 2012.

[103] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, p. 1087, 1953.

[104] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Springer, 1996.

[105] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[106] R. K. Morrow Jr. and J. S. Lehnert, "Bit-to-bit error dependence in slotted ds/ssma packet systems with random signature sequences," *IEEE Transactions on Communications*, vol. 37, no. 10, pp. 1052–1061, 1989.

[107] S. Ng, S. Leung, C. Chung, A. Luk, and W. Lau, "The genetic search approach. A new learning algorithm for adaptive IIR filtering," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 38–46, 1996.

[108] H.-N. Nguyen, D. Menard, R. Rocher, and O. Sentieys, "Energy reduction in wireless system by dynamic adaptation of the fixed-point specification," in *Proc. Conference on Design and Architectures for Signal and Image Processing (DASIP)*, Brussels, Belgium, Nov. 2008, pp. 132–139.

[109] H.-N. Nguyen, D. Menard, and O. Sentieys, "Design of optimized fixed-point WCDMA receiver," in *Proc. European Signal Processing Conference (EUSIPCO)*, Glasgow, UK, Aug. 2009, pp. 993–997.

[110] H.-N. Nguyen, D. Menard, and O. Sentieys, "Dynamic precision scaling for low power WCDMA receiver," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Taipei, Taiwan, May 2009, pp. 205–208.

[111] H.-N. Nguyen, D. Menard, and O. Sentieys, "Novel algorithms for word-length optimization," in *Proc. European Signal Processing Conference (EUSIPCO)*, Barcelona, Spain, Aug.–Sep. 2011, pp. 1944–1948.

[112] D. Novo, B. Bougard, A. Lambrechts, L. Van der Perre, and F. Catthoor, "Scenario-based fixed-point data format refinement to enable energy-scalable software defined radios," in *Proc. IEEE/ACM conference on Design, Automation and Test in Europe (DATE)*, Mar. 2008, pp. 722–727.

[113] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *Management Science*, pp. 797–813, 1996.

[114] G. Ochoa, I. Harvey, and H. Buxton, "Optimal mutation rates and selection pressure in genetic algorithms," in *Proc. Genetic and Evolutionary Computation Conference (GECCO)*. Las Vegas, Nevada, USA : Citeseer, 2000, pp. 315–322.

[115] W. Osborne, R. Cheung, J. Coutinho, W. Luk, and O. Mencer, "Automatic Accuracy-Guaranteed Bit-Width Optimization for Fixed and Floating-Point Systems," in *Proc. International Conference on Field Programmable Logic and Applications (FPL'07)*, 2007, pp. 617–620.

[116] E. Özer, A. Nisbet, and D. Gregg, "Stochastic Bit-width Approximation Using Extreme Value Theory for Customizable Processors," in *Internation Conference on Compiler Construction (CC)*. Springer, 2004, pp. 205–264.

[117] K. Parashar, R. Rocher, D. Menard, and O. Sentieys, "A Hierarchical Methodology for Word-Length Optimization of Signal Processing Systems," in *Proc. International Conference on VLSI Design*, Bangalore, India, Jan. 2010.

[118] J. Park, J. H. Choi, and K. Roy, "Dynamic Bit Width Adaptation in DCT : Image Quality versus Computation Energy Trade off," in *IEEE/ ACM conference on Design, Automation and Test in Europe 2006 (DATE 06)*, Munich, Germany, march 2006.

[119] J. Park, J. H. Choi, and K. Roy, "Dynamic Bit-width Adaptation in DCT : Image Quality versus Computation Energy Trade-off," in *Proc. IEEE/ACM conference on Design, Automation and Test in Europe (DATE)*, Munich, Germany, Mar. 2006.

[120] D. Pauluzzi and N. Beaulieu, "A comparison of SNR estimation techniques for the AWGN channel," *IEEE Transactions on Communications*, vol. 48, no. 10, pp. 1681–1691, Oct. 2000.

[121] M. Prais and C. Ribeiro, "Parameter variation in GRASP procedures," *Investigación Operativa*, vol. 9, pp. 1–20, 2000.

[122] J. G. Proakis, *Digital Communications*, 4th ed. McGraw-Hill, New York, 2000.

[123] M. Pursley, "Performance Evaluation for Phase-Coded Spread-Spectrum Multiple-Access Communication–Part I : System Analysis," *IEEE Transactions on Communications [legacy, pre-1988]*, vol. 25, no. 8, pp. 795–799, 1977.

[124] J. Radecki, J. Konrad, and E. Dubois, "Design of multidimensional finite-wordlength FIR and IIR filters by simulated annealing," *IEEE Transactions on Circuits and Systems—Part II : Analog and Digital Signal Processing*, vol. 42, no. 6, pp. 424–431, 1995.

[125] I. Rechenberg, "Cybernetic Solution Path of an Experimental Problem," *Library translation*, 1965.

[126] M. Resende and C. Ribeiro, "Greedy Randomized Adaptive Search Procedures," *International Series in Operations Research and Management Science*, pp. 219–250, 2003.

[127] M. Richey and H. Saiedian, "A New Class of Floating-Point Data Formats with Applications to 16-Bit Digital-Signal Processing Systems," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 95–101, Jun. 2009.

[128] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[129] R. Rocher, D. Menard, N. Herve, and O. Sentieys, "Fixed-Point Configurable Hardware Components," *EURASIP Journal on Embedded Systems*, vol. 2006, no. 1, pp. 20–20, 2006.

[130] R. Rocher, "Evaluation analytique de la précision des systèmes en virgule fixe," Ph.D. dissertation, Université de Rennes 1, 2006.

[131] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 96–101, 1994.

[132] M. P. S Dhanani, "Enabling high-precision dsp applications with the fpga industryís first variable-precision architecture," Altera inc., Tech. Rep., May 2010.

[133] C. Saiprasert, C. Bouganis, and G. Constantinides, "Word-Length Optimization and Error Analysis of a Multivariate Gaussian Random Number Generator," *Reconfigurable Computing : Architectures, Tools, and Applications (Proc. ARC'09)*, vol. 5453, pp. 231–242, Mar. 2009.

[134] J. D. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," in *Proc. International Conference on Genetic Algorithms (ICGA)*. L. Erlbaum Associates Inc., 1985, pp. 93–100.

[135] H. Schwefel, *Numerische Optimierung von Computer – Modellen mittels der Evolutionsstrategie*. Birkhäuser Basel, Stuttgart, 1977.

[136] H. Schwefel, "Collective phenomena in evolutionary systems," in *Preprints of the 31st Annual Meeting of the International Society for General System Research, Budapest*, vol. 2, no. S 1025, 1987, p. 1033.

[137] J. Skaf and S. Boyd, "Filter Design With Low Complexity Coefficients," *IEEE Transactions on Signal Processing*, vol. 56, pp. 3162–3169, 2008.

[138] B. Sklar, "Rayleigh fading channels in mobile digital communication systems. I. Characterization," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 136–146, 1997.

[139] N. Srinivas and K. Deb, "Multiobjective Optimization using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[140] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[141] W. Strauss, "DSP chips take on many forms," DSP-FPGA.com Magazine, March 2006.

[142] W. Strauss, "Hanging up on analog and flexing Wireless/DSP muscles," Forward Concepts, Tech. Rep., 2008.

[143] N. Sulaiman and T. Arslan, "A multi-objective genetic algorithm for on-chip real-time optimisation of word length and power consumption in a pipelined FFT processor targeting a MC-CDMA receiver," in *Proc. NASA/DoD Conference on Evolvable Hardware (EH)*, Washington, DC, 2005, pp. 154–159.

[144] W. Sung and K. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[145] K. Tachikawa, *W-CDMA Mobile Communications System*. Wiley, 2002.

[146] K. Tatas, G. Koutroumpezis, D. Soudris, and A. Thanailakis, "Architecture design of a coarse-grain reconfigurable multiply-accumulate unit for data-intensive applications," *Integration, the VLSI Journal*, vol. 40, no. 2, pp. 74 – 93, 2007, systems-on-Chip : Design and Test.

[147] V. T. Tovinakere, O. Sentieys, and S. Derrien, "Wakeup Time and Wakeup Energy Estimation in Power-Gated Logic Clusters," in *Proc. 24th International Conference on VLSI Design (VLSI Design)*, Chennai, India, 2011, pp. 340–345.

[148] M. Willems, V. Bursgens, and H. Meyr, "FRIDGE : Floating-Point Programming of Fixed-Point Digital Signal Processors," in *8th International Conference on Signal Processing Applications and Technology (ICSPAT 97)*, San Diego, US, September 1997.

[149] P. B. Wilson and M. D. Macleod, "Low implementation cost IIR digital filter design using genetic algorithms," in *Proc. IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, vol. 1, Nov. 1993, pp. 4/1–4/8.

[150] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[151] B. Wu, J. Zhu, and F. Najm, "An analytical approach for dynamic range estimation," in *Proc. ACM/IEEE Design Automation Conference (DAC)*, San Diego, US, Jun. 2004, pp. 472–477.

[152] T. Xanthopoulos and A. Chandrakasan, "A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 5, pp. 740–750, 2000.

[153] D. Xu and M. Daley, "Design of optimal digital filter using a parallel genetic algorithm," *IEEE Transactions on Circuits and Systems—Part II : Analog and Digital Signal Processing*, vol. 42, no. 10, pp. 673–675, 1995.

[154] M. Yannakakis, "The Analysis of Local Search Problems and Their Heuristics," in *Proc. 7th Annual Symposium on Theoretical Aspects of Computer Science (STACS'90)*. Springer-Verlag, 1990, pp. 298–311.

[155] S. Yoshizawa and Y. Miyanaga, "Tunable word length architecture for low power wireless OFDM demodulator," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2006, pp. 2789–2792.

[156] H. Zhao, T. Ottosson, E. Strom, and A. Kidiyarova-Shevchenko, "Performance analysis of a fixed-point successive interference canceller for WCDMA," in *Proc. IEEE Vehicular Technology Conference (VTC–Fall)*, vol. 3, Sep. 2004, pp. 1919–1923.

[157] R. Zimmermann, *Binary Adder Architectures for Cell-Based VLSI and their Synthesis*. HartungñGorre, 1998.

[158] E. Zitzler, "Evolutionary algorithms for multiobjective optimization : Methods and applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), 1999.

VU :

Le Directeur de thèse

Olivier Sentieys
Professeur
IRISA/ENSSAT

VU :

Le Responsable de l'École Doctorale

ECOLE DOCTORALE MATISSE
Le Directeur

DT URI 2011/ENSSAT/12 no 238

VU pour autorisation de soutenance

Rennes, le 7 décembre 2011

Le Président de l'Université de Rennes 1

Guy Cathelineau

P/le Président et par délégation
le Vice-Président

VU après soutenance pour autorisation de publication :

Le Président de Jury,

Résumé

Cette thèse s'inscrit dans le contexte de la forte augmentation du débit et de la puissance de calcul des systèmes de télécommunications. Cette augmentation entraîne une consommation d'énergie importante et réduit la durée de batterie, ce qui est primordiale pour un système embarqué. Nous proposons des mécanismes permettant de réduire la consommation d'énergie dans un système embarqué, plus particulièrement dans un terminal mobile sans fil.

L'implantation efficace des algorithmes de traitement numérique du signal dans les systèmes embarqués requiert l'utilisation de l'arithmétique virgule fixe afin de satisfaire des contraintes de coût, de consommation et d'encombrement. Dans les approches classiques, la largeur des données et des calculs est considérée au pire cas lors de la détermination des spécifications afin qu'elles soient satisfaites dans tout les cas. Nous proposons une approche d'adaptation dynamique, permettant de changer la spécification en fonction de l'environnement (par exemple les conditions d'un canal de transmission) avec pour objectif de réduire la consommation d'énergie dans certaines conditions. Tout d'abord, la relation entre la puissance de bruit de quantification et le taux d'erreur binaire du système en fonction du bruit au récepteur est établie pour une chaîne de transmission QPSK. Ce résultat est appliqué dans la technique d'accès multiple par répartition de codes en séquence directe (DS-CDMA). Parmi plusieurs systèmes de télécommunications utilisant la technique DS-CDMA, nous montrons comment adapter dynamiquement la précision de calcul d'un récepteur 3G WCDMA.

La conversion en virgule fixe nécessite un algorithme d'optimisation combinatoire pour l'optimisation des largeurs des opérateurs sous une contrainte de précision. La deuxième axe de ces travaux de thèse concerne l'étude d'algorithmes d'optimisation adaptés au problème de l'optimisation des largeurs de données. Nous proposons de nouveaux algorithmes pour les problèmes à une seule contrainte ou à une suite des contraintes correspondant à différents niveaux de précision pour les systèmes auto-adaptatifs. Le résultat des algorithmes génétiques multi-objectifs, sous forme d'une frontière de Pareto, permet d'obtenir la largeur correspondant à chaque niveau du bruit de quantification. Une version améliorée des algorithmes génétiques combinée avec l'élitisme et la recherche tabou est proposée. En plus, nous proposons d'appliquer GRASP, un algorithme de recherche locale stochastique permettant de trouver le résultat dans un temps plus faible en comparaison avec les algorithmes génétiques.

Abstract

Telecommunication systems today have increasing complexity and require higher energy consumption. Therefore, energy-efficiency becomes one of the most important goals in a design of embedded systems. This thesis proposes a dynamic scaling precision mechanism to reduce energy consumption in an embedded system, especially in a wireless mobile terminal.

Energy–efficient implementation of digital signal processing algorithms in embedded systems requires using fixed-point arithmetic in order to satisfy the cost, power and area constraints. In traditional approaches, the data bit-widths are calculated based on the worst case so that they are satisfied in all cases. We propose another approach to dynamically change the specification according to the environment (e.g. the transmission channel quality) with the goal of reducing energy consumption under certain conditions. Firstly, the relation between the quantization noise and the bit error rate at a receiver noise level is studied for a QPSK transmission system. The result is then applied in Direct Sequence Code Division Multiple Access (DS-CDMA) systems. Among several telecommunications systems using the DS-CDMA technique, we demonstrate how to dynamically adapt the fixed-point algorithm accuracy of a WCDMA 3G receiver.

The fixed-point conversion uses a combinatorial optimization algorithm for the determination of each bit-width under an accuracy constraint. The second part of this thesis concentrates on optimization algorithms. We propose new algorithms for problems having a single constraint or a series of constraints corresponding to different accuracy levels of a self-adaptive system. The result of Multi-Objective Genetic Algorithms (MOGA), a Pareto front, allows determining the bit-widths for each quantization noise level. An improved version of MOGA combined with elitism and tabu search is proposed. In addition, we suggest using GRASP, a stochastic local search algorithm, to find a result in a relatively short time in comparison with MOGA.