N° d'ordre : 13/2007-M/MT

République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE

FACULTE DES MATHEMATIQUES

MEMOIRE
Présenté à l'USTHB pour l'obtention du diplôme de

MAGISTER

En: Mathématiques
Spécialité : Recherche opérationnelle

Par

Lahlou BENGHEZAL

Sujet

Métaheuristiques pour le Recalage des Images en 2D, cas des Images Satellitaires

Soutenu publiquement le : 01 Octobre 2007 devant le jury composé de :

▪ M. BERRACHEDI A/Hafid	Professeur, l'USTHB	Président
• M. OUAFI Rachid	Maître de conférences, l'USTHB	Directeur de thèse
▪ M. BOUDHAR Mourad	Maître de conférences, l'USTHB	Examinateur
▪ M. AIT AOUDIA Samy	Maître de conférences, l'INI Oued Smar	Examinateur

2006-2007

Dédicaces

Je dédie ce travail

- A mes très chers parents ;
- A ma femme en particulier ;
- A mes enfants Chahinaze, Djazia et Ayoube Wassim ;
- A mes frères et sœurs ;
- A toute ma famille ;
- A tous mes amis ;
- A tous ceux qui me connaissent.

Remerciements

Je remercie tout particulièrement le promoteur M. Rachid OUAFI pour avoir accepté de m'encadrer, ainsi que pour ses nombreux conseils, suggestions et encouragements.

Je remercie M. BERRACHEDI A/Hafid Professeur à l'USTHB pour m'avoir fait l'honneur de présider le jury de magister.

Je remercie M. BOUDHAR Mourad Maître de conférences à l'USTHB et M. AIT AOUDIA Samy Maître de conférences à l'INI Oued Smar/El Harrach Alger d'avoir accepté d'être les examinateurs de cette thèse.

Merci également à tous les membres de l'équipe de la Recherche Opérationnelle de la Faculté des Mathématiques de l'USTHB B-EZZOUAR à Alger et de l'Institut National de la Cartographie et de la Télédétection (INCT H-Dey) pour la bonne ambiance qu'ils contribuent à créer, propice à la réflexion et à tous les invités et à toute l'aimable assistance.

Métaheuristiques pour le Recalage des Images en 2D
Cas des Images Satellitaires

Résumé

Un des problèmes les plus importants lors de traitement et d'analyse d'images satellitaires dans tous les domaines qui manipulent la télédétection est le recalage qui conditionne fortement l'exactitude et la cohérence des interprétations des images satellitaires. Ce dernier, étant la maximisation de similarité entre images, devient un problème d'optimisation. Nous pouvons donc envisager n'importe quelle méthode d'optimisation pour résoudre ce problème.

Notre objectif est d'étudier l'adéquation des métaheuristiques au problème de recalage, et de valider l'étude théorique par des tests expérimentaux sur des images satellitaires.

Nous avons développé un système pour le **recalage dense** basé sur la maximisation de similarité entre images par deux métaheuristiques, le **recuit simulé** (en codage naturel et binaire) et les **algorithmes génétiques** (en codage naturel, binaire et quantique).

Pour réduire le temps d'exécution du processus de recalage qui peut être prohibitif quand la taille des images est grande, nous avons adopté des schémas d'accélération basés sur la minimisation de la quantité d'informations à traiter et la parallélisation des méthodes d'optimisation.

Mots clés : *recalage, images satellitaires, métaheuristiques, algorithmes génétiques, recuit simulé, codage.*

Meta-heuristics for 2D Image Registration Case of Satellite Images

Abstract

One of the most important problems in satellite image processing and necessary to perform almost satellite imaging tasks in all field of the remote detection is the images registration which influences strongly the precision and the quality of satellite image interpretations. Images registration, as the maximisation of images likelihood, will be an optimization problem so we can use all optimization methods to resolve it.

Our objective is to prove the theoretical adequacy of the metaheuristics to the registration problem, and to validate the theoretical studies by performing some experimental tests on satellite images.

We have realized an intensity-based-registration-tool, which is based on the maximisation of images likelihood, by two metaheuristics: simulated annealing (natural & binary coding schemes) and genetics algorithms (natural, binary & quantum coding schemes).

We have also proposed acceleration schemes based on a data-quantity-minimization and parallel optimization methods.

Key-words: *registration, satellite images, metaheuristics, genetic algorithms, simulated annealing, coding.*

Sommaire

Introduction Générale **5**

Partie I. PROBLEMATIQUE DU RECALAGE DES IMAGES **8**

CHAPITRE I. Recalage des images satellitaires (état de l'art) **9**

I.1. Introduction 9
I.2. télédetection 10
I.2.1. Spectre électromagnétique : 11
I.2.2. Les interactions : 12
I.2.3. Détection passive et active : 12
I.3. images multispectrales : 13
I.4. Satellites et capteurs d'observation de la terre : 14
I.4.1. Landsat : 14
I.4.2. Spot : 15
I.5. Imagerie numérique 16
I.5.1. Le « Picture element » ou pixel 16
I.5.2. Caractéristiques d'une image mosaïque 17
I.6. Traitement d'images 18
I.6.1. Filtrage et réduction de bruit 18
I.6.2. Segmentation 18
I.6.3. Classification 18
I.6.4. Recalage d'images 18
I.7. Recalage des images (définitions et notations) 19
I.7.1. Problème de recalage 19
I.7.2. Modélisation Mathématique du problème 20
I.8. Préservation de topologie 20
I.9. Type des primitives considérées 21
I.9.1. Méthodes extrinsèques 21
I.9.2. Méthodes intrinsèques 21
I.10. Modalités et régions de recalage 23
I.10.1. Recalage monomodal d'une même région 23
I.10.2. Recalage multimodal d'une même région 23
I.10.3. Recalages monomodal/multimodal inter-régions 23
I.11. Elasticité de la transformation 24
I.11.1. Transformation rigide 25
I.11.2. Transformation affine 25
I.11.3. Transformation « projective » 25
I.11.4. Transformation déformable (ou non rigide) 25
I.12. Critères de similarité 26
I.12.1. Critères de similarité pour une approche par primitives géométriques 26
I.12.2. Critères de similarité pour une approche dense 26
I.12.3. Erreur quadratique: 26
I.12.4. Entropie: 26
I.12.5. Covariance : 27
I.12.6. Rapport de corrélation : 27
I.12.7. Information mutuelle : 27
I.13. Critère de similarité pour un processus de recalage 28
I.13.1. Recherche de la transformation optimale 28
I.13.2. Applications de la mise en correspondance 29
I.13.3. Formulation – Terminologie 29
I.14. Etat de l'Art du recalge 30
I.14.1. Recalage d'images satellitaires. 30
I.14.2. Recalage d'images non rigide. 30
I.14.3. Méthodes de recalage et de calibrage pour l'aide à l'interprétation d'images médicales en vraies couleurs 31

Partie II. Méthodes d'Optimisation pour le Recalage. 34

CHAPITRE II. Algorithmes génétiques 37
II.1. Origine des algorithmes génétiques 37
II.2. Description et définitions: 37
II.2.1. La population initiale 40
II.2.2. Mesurer la qualité (fonction de fitness) 40
II.2.3. Les opérateurs 40
II.2.4. Critère d'arrêt 45
II.2.5. Réglage des paramètres d'un algorithme génétique 45
II.2.6. Schéma de déroulement de l'algorithme génétique 46

CHAPITRE III. Recuit Simulé 47
III.1. origine du recuit simule 47
III.2. Description de l'opération physique « recuit »: 47
III.2.1. Critère de Metropolis 49
III.2.2. Caractéristiques du Recuit Simulé 50
III.2.3. Critère d'arrêt 52
III.2.4. Algorithme de recuit Simulé 52
III.3. Algorithme Liées au recuit simulé (Cas Continu) 53
III.3.1. L'algorithme CSA (Classical Simulated Annealing) 53
III.3.2. L'algorithme FSA (Fast Simulated Annealing) 54
III.3.3. L'algorithme ASA (Adaptive Simulated Annealing) 54

Partie III. Accélération et Parallélisation 56

CHAPITRE IV. Accélération sequentiel du recalage 57
IV.1. Introduction 57
IV.2. Diminuer la quantité d'informations à traiter 58
IV.3. Validation de la proposition 59
IV.3.1. Schéma d'accélération 60
IV.3.2. Tests de validation (problèmes dont on connaît la solution) 60
IV.4. Une méthode de recalage automatique 62
IV.5. Résultats apportés 63
IV.6. Conclusion 63

CHAPITRE V. Parallélisation du recalage 64
V.1. Introduction 64
V.2. Parallélisation des algorithmes génétiques 64
V.2.1. Parallélisation maître/esclave (GPGA, global master/slave parallel genetic algorithms) : 65
V.2.2. Parallélisation par sous populations statiques avec opérateur de migration 65
V.2.3. Les algorithmes génétiques hiérarchiques 67
V.2.4. Parallélisation par sous population sans opérateur de migration 67
V.2.5. Algorithme génétique massivement parallèle 67
V.2.6. Ilots ou sous populations dynamiques 67
V.3. Les architectures des algorithmes génétiques parallèles 68
V.4. L'Algorithme Génétique parallèle : 71
V.4.1. Conclusion sur le parallélisme des AGs: 71
V.5. Accélération de l'algorithme du recuit simulé 72
V.6. Le Recuit Simulé Evolutionnaire Parallèle (ESA) : 72
V.6.1. Le Recuit Simulé Evolutionnaire Parallèle sur un système multi agents: 73
V.6.2. Conception du système multi îlots : 74
V.6.3. Spécification des îlots : 74
V.7. Accélération par choix de paramètre 75
V.8. Conclusion 75

Partie IV Conception et mise en œuvre 76

CHAPITRE VI. Conception 77
VI.1. Introduction 77
VI.2. Conception 78
VI.2.1. Recalage par métaheuristiques 79
VI.2.2. Recalage par algorithmes génétiques 80
VI.2.3. Recalage par recuit simulé 94
VI.2.4. Recalage par recuit simulé (en codage binaire) 99
VI.3. Conclusion 99

CHAPITRE VII. Mise en œuvre 100
VII.1. Introduction 100
VII.2. Développement du logiciel «RIS-MH» 100
VII.3. Compilateur utilisé 101
VII.4. Noyau élémentaire 101
VII.5. Mise en œuvre des métaheuristiques 107
VII.5.1. Mise en œuvre des méthodes « algorithmes génétiques » 107
VII.5.2. Mise en œuvre des méthodes « recuit simulé » 109
VII.6. Schéma conceptuel Réduit du noyau en mode séquentiel 115
VII.7. Schéma conceptuel Réduit du noyau en mode parallèle 116
VII.8. Conclusion 117

CHAPITRE VIII. Tests et résultats 118
VIII.1. Introduction 118
VIII.2. Définitions 118
VIII.3. Méthodologie des tests 119
VIII.3.1. Choix des paramètres optimaux 119
VIII.4. Validation du choix des paramètres 123
VIII.5. Exécution des algorithmes génétiques 123
VIII.5.1. Réglage des paramètres génétiques 123
VIII.5.2. Taille de l'espace de recherche et qualité de la recherche 123
VIII.5.3. Taille des populations, qualité de recalage et le surcoût 124
VIII.5.4. L'impact des l'opérateur de croisement 127
VIII.5.5. Etude des tournois binaires 128
VIII.5.6. Choix du critère de similarité 131
VIII.5.7. Codage des individus 132
VIII.5.8. Les images multimodales 134
VIII.6. Exécutions de la méthode recuit simulé 135
VIII.6.1. Calibrage des paramètres recuit simulé 135
VIII.6.2. Taille de l'espace des voisins 135
VIII.6.3. Rapidité de refroidissement (Alpha) 137
VIII.6.4. Le nombre des étapes 137
VIII.6.5. Le nombre des itérations par étape 139
VIII.6.6. Fonction de voisinage 140
VIII.6.7. Type du codage des solutions 141
VIII.6.8. Tests des paramètres sélectionnés sur d'autres problèmes pour valider le choix 142
VIII.6.9. Problèmes réels 142
VIII.7. Comparaison « recuit simulé Vs algorithmes génétiques » 143
VIII.7.1. Réglage des paramètres 143
VIII.8. Synthèse des résultats 144
VIII.9. Autres tests 145
VIII.10. Tests & résultats 146
VIII.10.1. Exemple 1 146
VIII.10.2. Exemple 2 147
VIII.10.3. Exemple 3 148
VIII.10.4. Exemple 4 149
VIII.11. Conclusion 150

Conclusion générale & perspectives 152

Liste des figures 154

Liste des tableaux 156

Liste des algorithmes 157

Liste des équations 158

Bibliographie 174

Annexe I : L'Accélération de calcul & L'Informatique Quantique. 160

Annexe II : Satellite Images Registration by Metaheuristics. 168

Introduction générale

INTRODUCTION GENERALE

Un problème d'optimisation combinatoire est un problème qui peut s'exprimer par une fonction (dite de coût) avec ou sans contraintes, à minimiser ou maximiser sur un ensemble de définition fini ou dénombrable. C'est le cas de nombreux problèmes, dans des domaines d'applications très variés, qu'ils soient scientifiques ou techniques. Pour illustrer ce propos on peut citer des problèmes académiques tels que la coloration de graphes ou le sac à dos multidimensionnel, ainsi que des applications réelles comme le positionnement de composants dans la conception de circuits imprimés, la définition de réseaux de radio émetteurs ou la restauration des images. Aussi, l'optimisation combinatoire est un domaine qui fait l'objet de recherches intenses.

Tous les problèmes d'optimisation n'ont bien entendu pas le même degré de difficulté, celui ci étant surtout lié à la dimension de l'espace de recherche et au "paysage" de la fonction à optimiser (nombre de minima, dérivabilité, etc.). En conséquence, de multiples algorithmes d'optimisation ont été développés, certains étant plus adaptés à des problèmes présentant certaines caractéristiques, tandis qu'ils échouent sur des problèmes où d'autres méthodes d'optimisation sont adéquates, et inversement. Ces algorithmes d'optimisation peuvent être classés de différentes manières. Nous distinguons ainsi les classes suivantes algorithmes d'optimisation locale, algorithmes d'optimisation globale. Alors que les algorithmes de la première classe sont piégés par le premier minimum qu'ils rencontrent ou sont handicapés par la taille de l'espace de recherche, les algorithmes de la seconde classe ne présentent pas ces inconvénients et permettent de trouver une solution « proche » de l'optimum global. En revanche, les algorithmes de la première classe convergent plus rapidement que ceux de la seconde, tout en ayant un coût calculatoire moindre. En définitive, il n'existe pas de meilleur algorithme d'optimisation en termes de performances, que ce soit au niveau de la qualité des résultats ou des temps de calcul, et ceci indépendamment du problème considéré. Le travail que nous présentons a essentiellement deux objectifs. D'une part l'étude d'algorithmes d'optimisation globale dans le cadre d'un problème particulier, d'autre part l'évaluation de l'adéquation de ces algorithmes à certains modes de parallélisations.

Nous avons choisi comme champ d'étude le recalage en imagerie satellitaire, plus précisément le recalage rigide et le recalage déformable (tout deux denses, i.e. basés sur les niveaux de gris, en 2D). Dans ce contexte, le but du recalage est la mise en correspondance des structures géographiques afin de suivre, par exemple, l'évolution de constructions ou d'évaluer l'ampleur d'un désastre. De nombreuses méthodes de recalage en imagerie satellitaire consistent à minimiser une fonction de coût, ou fonction de similarité exprimant la similitude au niveau des pixels en 2D des images que l'on cherche à recaler entre elles. Or les fonctions de similarité classiques en recalage des images sont non linéaires, irrégulières et présentent de nombreux minima locaux. Pour résoudre ce problème d'optimisation difficile, il est donc nécessaire de recourir à un algorithme d'optimisation globale.

La première partie est consacrée à l'introduction générale à la problématique du recalage en imagerie satellitaire 2D mono et multimodale. Un état de l'art succinct permet au lecteur de se faire une idée de la difficulté posée par le recalage dans le domaine de la télédétection. Nous décrivons ensuite plus précisément les deux problèmes de recalage que nous avons retenus, à savoir le recalage rigide et le recalage déformable. La transformation rigide permet de faire un recalage global, tandis que la transformation déformable prend en compte des variations qui sont locales.

Dans la seconde partie, nous faisons une présentation de différents algorithmes d'optimisation globale divers aspects de chaque algorithme sont abordés, en particulier sur le plan théorique lorsqu'un modèle mathématique permettant de modéliser l'algorithme existe (preuve de la convergence, choix des paramètres, etc.).

Dans la troisième partie, on s'intéresse à l'accélération du processus et la parallélisation des méthodes d'optimisation, nous avons essayé de faire le panorama le plus complet possible.

Dans la quatrième partie, on montre le travail qu'on a mené par la conception et mise en œuvre des algorithmes d'optimisation et d'évaluation et leurs implantations sur machine pour les traitements à l'aide de l'outil informatique et faire les expériences pratiques de recalage, nous réalisons des tests d'évaluation sur des images satellitaires et nous effectuons une comparaison des résultats et une évaluation des méthodes de résolution.

Les problèmes d'optimisation combinatoire suscitent beaucoup d'intérêts. Malgré les progrès considérables de l'outil informatique, les méthodes d'énumération, exhaustive ou partielle, sont encore peu satisfaisantes en temps d'exécution ou en efficacité. Comme ces problèmes contiennent souvent beaucoup de solutions à intérêts pratiques acceptables, les spécialistes de l'optimisation combinatoire ont orienté leur recherche vers le développement des méthodes heuristiques. Le but est de trouver une solution de qualité satisfaisante en un temps de calcul raisonnable.

D'autant plus que pour des problèmes réels, il n'est pas toujours impératif de trouver la solution optimale, mais des solutions dont la qualité et le temps pour l'obtenir restent dans l'acceptable. Ces performances étant de nature opposée, il s'agit alors de trouver un compromis selon le contexte du problème.

Les objectifs de ce mémoire de magister sont essentiellement l'étude d'algorithmes d'optimisation globale dans le cadre d'un problème particulier qui est le recalage des images satellitaires, et d'autre part l'accélération de ces algorithmes et on se propose, pour l'amélioration des calculs trop élevé, la parallélisation des méthodes d'optimisation sur une architecture dite généraliste constituée d'un réseau performant de PC sera développée pour une mise en œuvre en perspective.

- **Contribution de ce mémoire**

L'objet de ce Mémoire porte sur l'étude de deux méthodes heuristiques et une approche algorithmique pour résoudre le problème du recalage des images satellitaires. Le but final est de constituer une bibliothèque d'algorithmes à utiliser selon le choix de l'utilisateur (Choix des méthodes, critère d'optimalité, temps de calcul...). Des analyses des résultats et les comportements des différents algorithmes permettront de conclure sur l'efficacité de chaque algorithme en fonction des critères à optimiser.

Le problème particulier à étudier est le recalage des images 2D. Le recalage permet de faire, dans le cas des images satellitaires et/ou aériennes, la fusion des images et la mise en correspondance des zones géographiques afin de déceler par exemple l'évolution de l'environnement, l'évaluation des changements et l'analyse thématique (ex des nouvelles constructions, de la verdure ou la sécheresse et détecter les différents mouvements des engins). De nombreuses méthodes de recalage en imagerie consistent à minimiser une fonction de coût exprimant la similitude au niveau des pixels en 2D des images que l'on cherche à recaler entre elles.

Ces fonctions sont généralement non linéaires, irrégulières et présentent de nombreux minima locaux, d'où l'utilité du recours aux algorithmes d'optimisation globale.

L'étude concernera l'implémentation de plusieurs méthodes d'optimisation qui sont les algorithmes génétiques, le recuit simulé et pour pallier au coût élevé des calculs dans le cas de l'application du recalage, on se propose d'accélérer les méthodes d'optimisation.

PARTIE I. PROBLEMATIQUE DU RECALAGE DES IMAGES.

CHAPITRE I. RECALAGE DES IMAGES SATELLITAIRES (ETAT DE L'ART)

I.1. INTRODUCTION

Le recalage (ou la mise en correspondance) des images est un problème d'optimisation car il se traduit par l'optimisation d'une certaine fonction, donc on peut utiliser des méthodes d'optimisation pour le résoudre, mais bien avant il faut bien étudier les caractéristiques (surtout la complexité) du recalage afin de bien choisir une méthode adéquate.

Pratiquement, le recalage des images est une tache très précieuse, dans le sens où plusieurs applications de vision par ordinateur font l'intervenir. L'application de la mise en correspondance dans le cadre d'imagerie satellitaire, importe un bénéfice important à cause du nombre important des taches de diagnostique et/ou thématique même de recherche complexe, qui devient très évidentes en cas d'utilisation du recalage. Citons par exemple la fusion des images satellitaires et/ou aériennes qui permet de localiser avec précision des éventuels changements, l'approche manuelle peut être fastidieuse (sans parler de la précision). Ce qui explique le nombre important des projets lancés et les articles publiés dans ce sens.

Toujours dans le cadre de l'imagerie satellitaire et comme on le verra plus tard, le recalage peut être utilisé pour une simple mise en même échelle ou une fusion de données qui ne nécessitent qu'une simple transformation avec peut de variables, à la création des atlas géographiques ou des cartographies opérationnelles qui nécessitent des transformations plus complexes dont le nombre de variables est plus important. Ce qui régie la variation de complexité des recalages, dans ce qui suit on va essayer donc à énumérer les paramètres qui contrôlent la complexité d'une méthode de recalage, et ensuite d'essayer de les classées.

Pour interagir, et surtout comprendre ce qui se passe autours de lui, l'être humain utilise ses cinq sens. D'après les psychos physiologistes, la vue est le sens primordial pour l'être humain, le système de vision humain derrière l'œil est, donc, le moyen de communication et d'interaction le plus efficace et le plus important. Cette importance n'est pas seulement due à la puissance et à la complexité du système de vision humain, mais aussi à la richesse des images en terme d'informations portées par elles.

La vision par ordinateur était parmi les premières voies proposées avec la révolution de l'informatique, en réalité, les ordinateurs deviennent de plus en plus puissants. Cette vision par ordinateur n'avait pas de sens sans une définition de l'image compatible avec les principes numériques qui caractérisent les informations manipulées par l'ordinateur. Cette définition ne pouvait se faire sans un processus de numérisation d'images, qui sera vue comme étant « la projection bidimensionnelle contenant des points caractérisés par l'intensité lumineuse perçue par un capteur quelconque », cette unité s'appelle le pixel, qui devient l'unité de base pour la construction d'images.

I.2. TELEDETECTION

L'information géographique est une denrée de plus en plus précieuse, que ce soit à des fins commerciales (exploitation des ressources terrestres), stratégiques (surveillance des territoires) ou humaines (aide au développement, prévention des catastrophes). Depuis un siècle on a su « prendre de la hauteur » pour observer la terre à bord de ballons, fusées, avions, hélicoptères, drones, satellites et navettes spatiales. La télédétection est l'ensemble des techniques qui permettent d'acquérir à distance des informations, en général des images, et de les traiter pour répondre à des problèmes.

La télédétection est la technique [ROB02] qui, par l'acquisition d'images, permet d'obtenir de l'information sur la surface de la terre sans contact direct avec celle-ci. La télédétection englobe tout le processus qui consiste à capter et à enregistrer l'énergie d'un rayonnement électromagnétique émis ou réfléchi, à traiter et à analyser l'information, pour ensuite mettre en application cette information.

Dans la plupart des cas, la télédétection implique une interaction entre l'énergie incidente et les cibles. Le processus de la télédétection au moyen de systèmes imageurs comporte les sept étapes que nous élaborons ci-après.

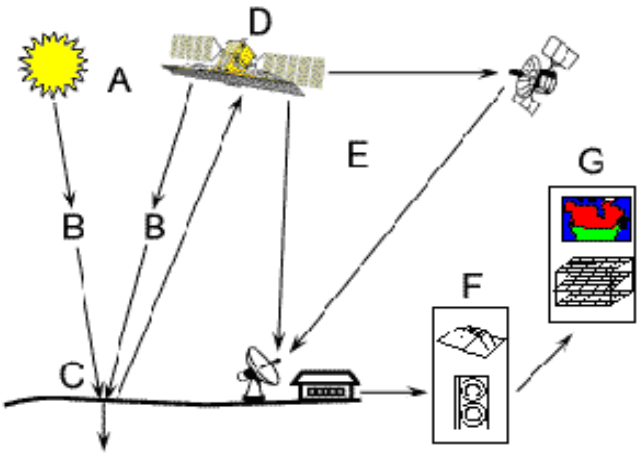


Figure 1. Les étapes de la télédétection.

1. Source d'énergie ou d'illumination (A) - À l'origine de tout processus de télédétection se trouve nécessairement une source d'énergie pour illuminer la cible.

2. Rayonnement et atmosphère (B) - Durant son parcours entre la source d'énergie et la cible, le rayonnement interagit avec l'atmosphère. Une seconde interaction se produit lors du trajet entre la cible et le capteur.

3. Interaction avec la cible (C) - Une fois parvenue à la cible, l'énergie interagit avec la surface de celle-ci. La nature de cette interaction dépend des caractéristiques du rayonnement et des propriétés de la surface.

4. Enregistrement de l'énergie par le capteur (D) - Une fois l'énergie diffusée ou émise par la cible, elle doit être captée à distance (par un capteur qui n'est pas en contact avec la cible) pour être enfin enregistrée.

5. Transmission, réception et traitement (E) - L'énergie enregistrée par le capteur est transmise, souvent par des moyens électroniques, à une station de réception où l'information est transformée en images (numériques ou photographiques).

6. Interprétation et analyse (F) - Une interprétation visuelle et/ou numérique de l'image traitée est ensuite nécessaire pour extraire l'information que l'on désire obtenir sur la cible.

7. Application (G) - La dernière étape du processus consiste à utiliser l'information extraite de l'image pour mieux comprendre la cible.

I.2.1. Spectre électromagnétique :

Les capteurs utilisés en télédétection sont des radiomètres imageurs. En effet, ils mesurent des rayonnements et organisent ces mesures sous forme d'images. Ces images sont utilisées pour obtenir des informations sur les objets qu'elles représentent. Or, le seul lien qui relie l'image à l'objet est le rayonnement émis ou réfléchi par cet objet et reçu par le radiomètre. La télédétection s'appuie donc avant tout sur une bonne connaissance des rayonnements électromagnétiques et de leur comportement au contact de la terre et à travers l'atmosphère.

Le rayonnement électromagnétique est un transport d'énergie dans l'espace sous la forme d'une onde caractérisée par un champ magnétique et un champ électrique étroitement associés, ou de particules correspondantes.

Les différentes régions du spectre se caractérisent par la longueur d'onde, mais aussi par leur fréquence. La fréquence est inversement proportionnelle à la longueur d'onde. Le spectre est divisé en plusieurs catégories. Par longueur d'onde décroissante : ondes radio, hyperfréquences, infrarouge, lumière visible, Ultraviolet, rayons X et rayons γ. Le spectre électromagnétique, qui regroupe toutes les catégories de lumière, est présenté par la figure suivante :

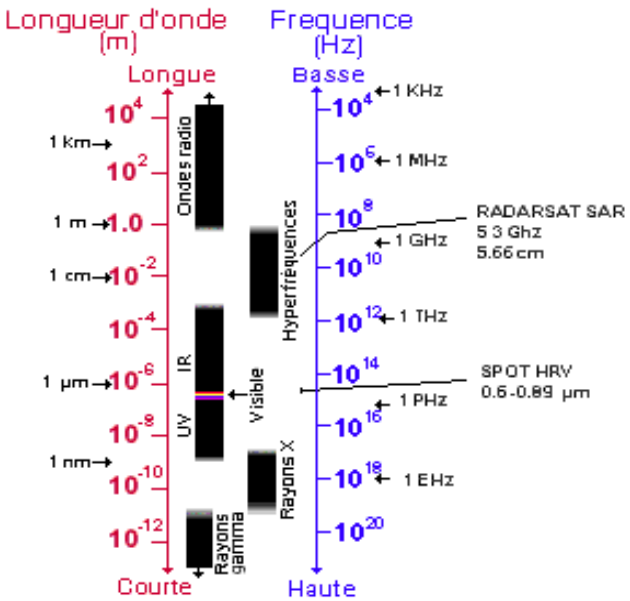


Figure 2. Le spectre électromagnétique

I.2.2. Les interactions :

Il existe deux types d'interactions des rayonnements, avec l'atmosphère et avec la terre.

I.2.2.1. Interactions avec l'atmosphère :

Avant que le rayonnement utilisé pour la télédétection n'atteigne la surface de la terre, celui-ci doit traverser une certaine épaisseur d'atmosphère. Les particules et les gaz présents dans l'atmosphère peuvent dévier ou bloquer le rayonnement incident. Ces effets sont causés par les mécanismes de diffusion et d'absorption. La diffusion se produit lors de l'interaction entre le rayonnement incident et les particules ou les grosses molécules de gaz présentes dans l'atmosphère. Les particules dévient le rayonnement de sa trajectoire initiale. Le niveau de diffusion dépend de plusieurs facteurs comme la longueur d'onde, la densité de particules et de molécules, et l'épaisseur de l'atmosphère que le rayonnement doit franchir.

I.2.2.2. Interaction avec la surface terrestre :

La réflexion d'une onde incidente sur la surface terrestre dépend des caractéristiques de l'onde (longueur d'onde, polarisation, angle d'incidence, etc.) et des caractéristiques de la surface (matériau, pente, rugosité, humidité, etc.). En fonction de ces caractéristiques la réflexion peut être spéculaire (effet de miroir) ou diffuse.

Figure 3. La réflexion terrestre.

I.2.3. Détection passive et active :

I.2.3.1. Capteurs passifs :

L'énergie du Soleil est soit réfléchie (la portion visible) ou absorbée et retransmise (infrarouge thermique) par la cible. Les dispositifs de télédétection qui mesurent l'énergie disponible naturellement sont des capteurs passifs. Le capteur passif peut seulement percevoir l'énergie réfléchie lorsque le Soleil illumine la terre. Il n'y a donc pas d'énergie solaire réfléchie le soir, tandis que l'énergie dégagée naturellement (l'infrarouge thermique) peut être perçue le jour ou la nuit.

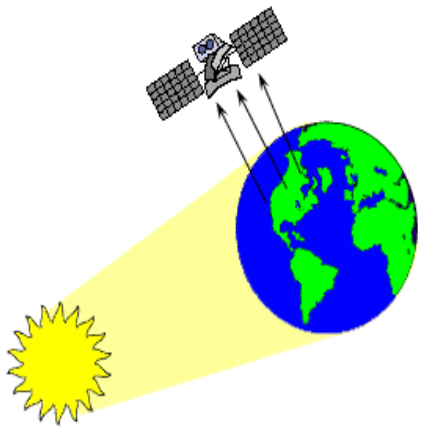


Figure 4. capteur passif

I.2.3.2. Capteurs actifs :

Ils Produisent leur propre énergie pour illuminer la cible : ils dégagent un rayonnement électromagnétique qui est dirigé vers la cible. Le rayonnement réfléchi par la cible est alors perçu et mesuré par le capteur. Le capteur actif a l'avantage de pouvoir prendre des mesures à n'importe quel moment de la journée ou de la saison. Les capteurs actifs utilisent les longueurs d'onde qui ne sont pas produites en quantité suffisante par le Soleil telles que les hyperfréquences. Ils doivent produire une énorme quantité d'énergie pour bien illuminer une cible. Le laser fluoromètre et le radar à synthèse d'ouverture (RSO) sont des exemples de capteurs actifs.

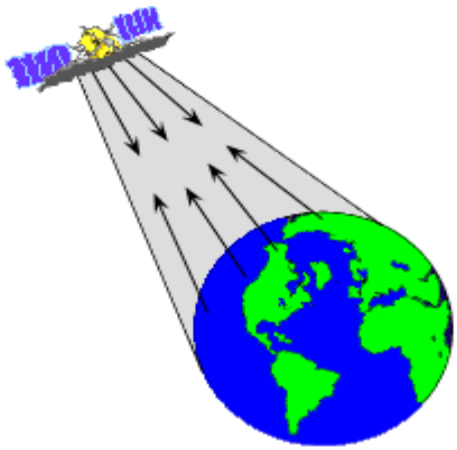


Figure 5. Capteur actif.

I.3. IMAGES MULTISPECTRALES :

Une image multi spectrale est un ensemble de matrices [ROB02] (une matrice par longueur d'onde) où sont enregistrées les mesures associées à chaque pixel (Figure 6) [KNIP,95], elle est acquise par un capteur caractérisée par un certain nombre de bandes de 1 à 200, dont chacune correspond à une longueur d'onde.

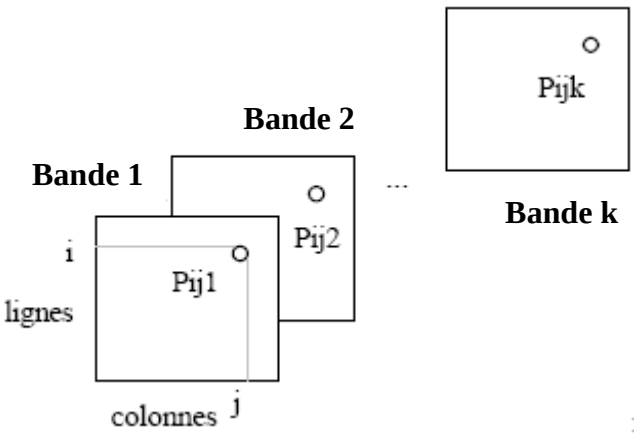


Figure 6. Image multi spectrale

Les images multi spectrales contiennent une information riche, qui permet de simplifier et de fiabiliser la détection et l'identification d'objets de faible taille et/ou de faible contraste.

I.4. SATELLITES ET CAPTEURS D'OBSERVATION DE LA TERRE :

I.4.1. Landsat :

Bien que plusieurs satellites soient également utilisés pour la surveillance de la surface de la terre, ceux-ci n'ont pas été conçus pour la cartographie détaillée de la surface terrestre. Le premier satellite d'observation Landsat-1 a été lancé par la NASA en 1972. Connu à l'origine sous l'acronyme ERTS-1 (Earth Resources Technology Satellite), Landsat avait été conçu pour tester la faisabilité d'une plate-forme multi spectrale d'observation de la terre non habitée. Depuis, le programme Landsat a permis l'acquisition de données sur tous les coins de la planète. En 1985, le programme a été commercialisé pour fournir des données aux divers utilisateurs civils.

Les trois premiers satellites (Landsat-1 à Landsat-3) se situaient à une altitude de 917 km avec une répétitivité de 18 jours, tandis que les derniers orbitent à une altitude approximative de 705 km avec une répétitivité de 16 jours.

Les satellites de la série Landsat portent plusieurs capteurs comme les systèmes de caméras RBV (Return Beam Vidicon), le système MSS (Multi Spectral Scanner), le TM (Thematic Mapper) qui a commencé avec Landsat 4.

Le capteur TM a apporté plusieurs améliorations par rapport à ses prédécesseurs : une meilleure résolution spatiale de 30 m pour toutes les bandes, des bandes spectrales plus étroites (au nombre de sept). Le tableau suivant décrit la résolution spectrale des bandes individuelles TM et leurs applications.

Bandes TM :

Bandes	Domaine spectral (microns)	Application
TM 1	0,45 - 0,52 (bleu)	discrimination entre le sol et la végétation, bathymétrie/ cartographie côtière; identification des traits culturels et urbains
TM 2	0,52 - 0,60 (vert)	cartographie de la végétation verte (mesure le sommet de réflectance); identification des traits culturels et urbains
TM 3	0,63 - 0,69 (rouge)	discrimination entre les espèces de plantes à feuilles ou sans feuilles; (absorption de chlorophylle); identification des traits culturels et urbains
TM 4	0,76 - 0,90 (proche infrarouge)	identification des types de végétation et de plantes; santé et contenu de la masse biologique; délimitation des étendues d'eau; humidité dans le sol

TM 5	1,55- 1,75 (infrarouge de courte longueur d'onde)	sensible à l'humidité dans le sol et les plantes; discrimination entre la neige et les nuages
TM 6	10,4- 12,5 (infrarouge thermique)	discrimination du stress de la végétation et de l'humidité dans le sol relié au rayonnement thermique; cartographie thermique
TM 7	2,08- 2,35 (infrarouge de courte longueur d'onde)	discrimination entre les minéraux et les types de roches; sensibles au taux d'humidité dans la végétation

Tableau 1. Bandes TM de Landsat.

Les données des capteurs TM sont utilisées pour plusieurs applications comme la gestion des ressources, la cartographie, la surveillance de l'environnement et la détection du changement.

I.4.2. Spot :

Le système SPOT (Satellite pour l'observation de la terre) est une série de satellites d'observation de la terre qui ont été conçus et lancés par le Centre National d'Études Spatiales (CNES) de la France, avec l'aide de la Belgique et de la Suède. SPOT-1 a été lancé en 1986, et a été suivi d'autres satellites lancés tous les trois ou quatre ans. Tous les satellites ont une altitude de 822 km, ce qui produit une répétitivité de 26 jours.

Tous les satellites SPOT ont deux balayeurs multi spectraux HRV (haute résolution visible) à barrettes, qui peuvent opérer indépendamment ou simultanément. Chaque HRV peut capter en mode panchromatique (une seule bande) et offre une excellente limite de résolution spatiale de 10 m. Ils peuvent aussi capter en mode multi spectral (MLA) (trois bandes) qui offre une résolution spatiale de 20 m. Le tableau suivant décrit les caractéristiques spectrales des deux modes.

Mode / bande		**Domaine spectral (microns)**
Panchromatique (PLA)		0,51 - 0,73 (bleu -vert- rouge)
Multi spectral (MLA)	Bande 1	0,50 - 0,59 (vert)
	Bande 2	0,61 - 0,68 (rouge)
	Bande 3	0,79 - 0,89 (proche infrarouge)

Tableau 2. Bandes de SPOT.

Le système SPOT présente plusieurs avantages par rapport aux autres capteurs spatiaux. Les données de trois bandes multi spectrales sont utiles pour afficher des images fausses-couleurs et la bande panchromatique peut être utilisée pour améliorer le détail des données multi spectrales.

I.5. IMAGERIE NUMERIQUE

I.5.1. Le « Picture element » ou pixel

Le pixel est l'élément de base pour la construction d'images. Il porte l'information sur la luminosité d'un point dans une image. C'est la combinaison de ces points avec leurs diverses valeurs de luminosités qui vont définir l'ensemble de toutes les images de l'univers. Ce modèle, facilitera l'exploitation et la manipulation efficace des informations contenues dans les images par l'ordinateur.

La manière d'exprimer la valeur de l'intensité d'un pixel créera différents types d'images, deux valeurs sont considérées, noir (0) et blanc (1). Les images à base de ce genre de pixels sont dites monochromes. Le pixel est codé sur 1 bit.

Plusieurs valeurs sont représentées, et vont modéliser le passage progressif entre les deux couleurs noire et blanche à travers le gris. Généralement nous avons 256 valeurs (1 octet), et les images à base de cette modélisation sont dites de niveaux de gris.

Nous avons rapidement constaté, qu'on peut obtenir toutes les couleurs avec un mélange de trois couleurs de base seulement le rouge, le vert et la couleur bleue (RVB). La modélisation RVB consiste à représenter la couleur d'un pixel avec le mélange des trois couleurs, le codage nécessite 3 octets (un par couleur de base), et ainsi on peut obtenir des images en couleurs.

Une image est une représentation dans le plan d'un objet réel. Les images sont omniprésentes dans notre société (dessins, peintures, photographies, etc.). Depuis quelques années, on assiste à une invasion d'images numériques, aussi appelées images "digitales" par emprunt à l'Anglais ("digit" = nombre).

Ces images sont constituées d'un très grand nombre de très petits points dont la couleur (ou l'intensité) est définie par des chiffres. Ces points sont appelés "pixels", dérivé de l'anglais "Picture éléments".

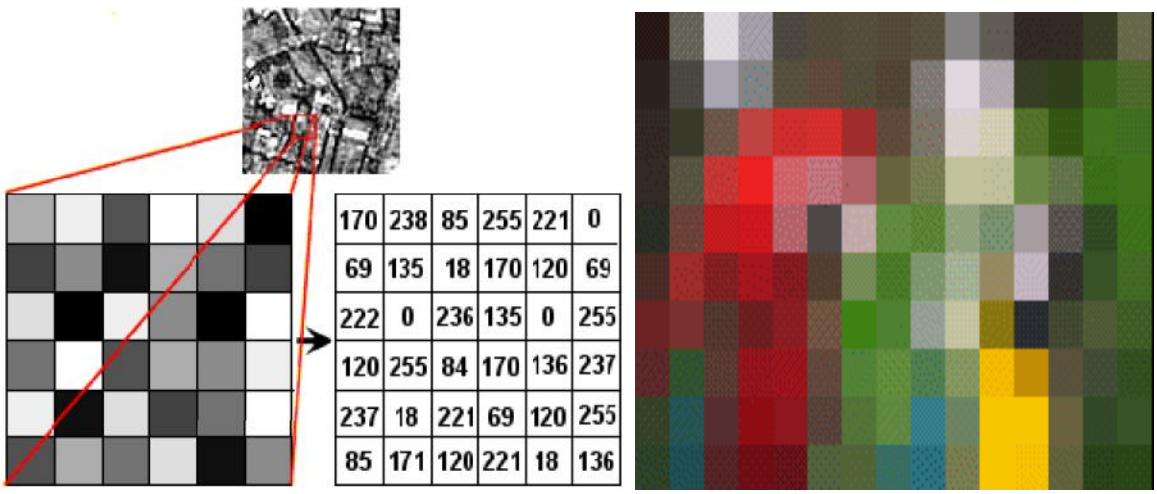


Figure 7. Les images numériques

I.5.2. Caractéristiques d'une image mosaïque

On appelle image mosaïque les images à base de pixels, ce genre d'images a les propriétés suivantes

I.5.2.1. La résolution d'une image

Etant une projection bidimensionnelle d'une scène, l'image est vue comme une matrice de pixels. Le nombre de lignes et le nombre de colonnes définissent la résolution de l'image.

I.5.2.2. Histogramme d'une image

C'est un graphique bidimensionnel indiquant la pertinence des niveaux de gris dans une image. Sur l'axe des X sont alignés toutes les valeurs des niveaux de gris, et sur l'axe des Y le nombre d'occurrences des niveaux de gris.

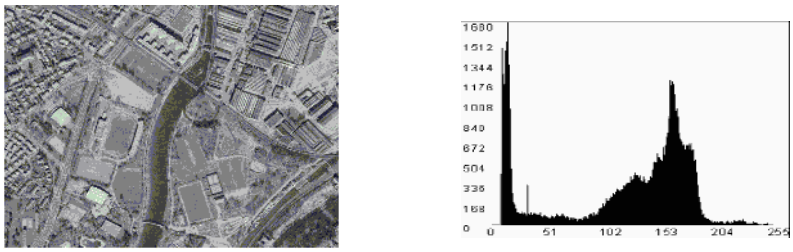


Figure 8. Une image et son histogramme.

I.5.2.3. Relation de voisinage entre pixels

Le voisinage d'un pixel est l'ensemble des pixels qui l'entourent immédiatement.

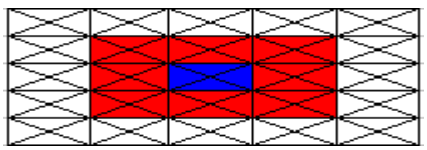


Figure 9. Voisinage (en rouge) d'un pixel (en bleue).

I.5.2.4. Régions et contours

Une région c'est l'ensemble des pixels voisins qui ont des valeurs des niveaux de gris homogènes, un contour est la frontière définie par une région.

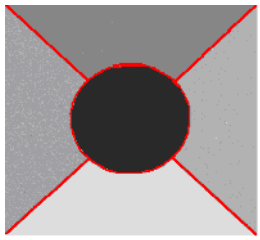


Figure 10.Régions délimitées par des contours (en rouge).

I.6. TRAITEMENT D'IMAGES

On appelle traitement d'images l'ensemble des processus permettant l'exploitation et/ou la manipulation des images. A titre d'exemple, un traitement de l'image peut être

I.6.1. Filtrage et réduction de bruit

Filtrer l'image c'est corriger les valeurs des niveaux de gris de certains pixels, ces pixels créent avec leurs voisins des passages brusques des niveaux de gris.

I.6.2. Segmentation

La segmentation consiste à découper une image en un ensemble de régions homogènes et connexes, le regroupement de ces régions doit reconstruire l'image d'origine. La segmentation d'image est l'une des tâches les plus importantes en vision artificielle, c'est une tâche de bas niveau nécessaire à rendre possible (ou faciliter) d'autres applications.

I.6.3. Classification

C'est de ranger l'ensemble des primitives géométriques (pixels, régions, …etc.) d'une image dans les classes ou les catégories selon un critère particulier. Plusieurs méthodes ont été proposées pour accomplir cette tâche, ces méthodes sont classées selon l'approche considérée ou le type de primitives traitées.

I.6.4. Recalage d'images

Contrairement aux tâches décrites précédemment qui ne traitent qu'une seule image, le recalage manipule deux ou plusieurs images. Le but d'un recalage est de mettre en correspondance les informations communes des images, moyennant des transformations. Des explications détaillées sont décrites au chapitre suivant.

I.7. RECALAGE DES IMAGES (DEFINITIONS ET NOTATIONS)

Le recalage (ou encore la mise en correspondance) peut se faire entre plusieurs images dites « à recaler » sur une autre image dite « de référence ou consigne », mais pour simplifier on admettra que le recalage se fait entre une image à recaler Ir et une image consigne Ic.

Et donc, le recalage d'une image sur une image de référence, est l'alignement de l'image à recaler de sorte que les structures exprimant les mêmes informations soient superposées dans les deux images. Plus précisément, c'est l'optimisation (plus souvent minimisation) d'une fonction exprimant la similarité entre les deux images : image de référence et image recalée [ZIT03].

I.7.1. Problème de recalage

On peut voir la procédure de recalage comme étant un système représenté par une boite noire constituée des éléments suivants :

- **En entrée** (essentiellement) une image consigne, et une image à recaler.
- **En sortie** l'image recalée dont la similarité avec l'image consigne est maximale, c'est la fonction objective definie comme suit :

$$\text{MAX Similarité}\,(Ic, \mathrm{T}\,[Ir]).$$

Equation 1. Fonction objective.

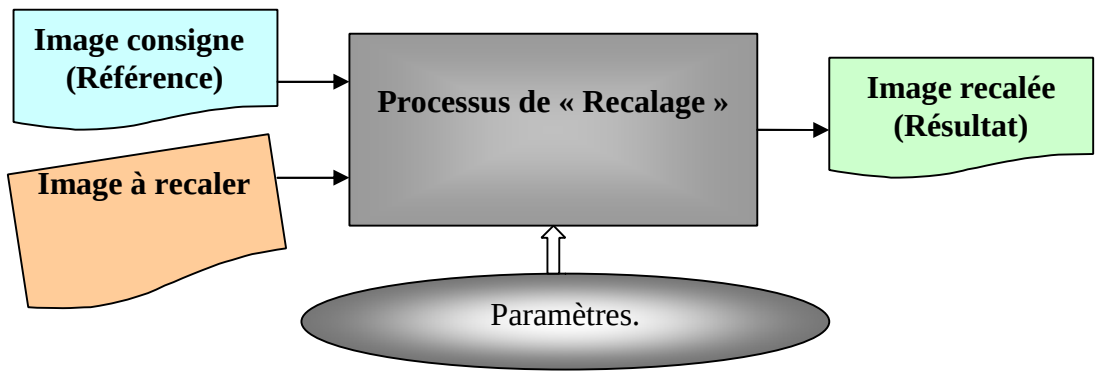


Figure 11. Entrées et sorties de la procédure de recalage.

I.7.2. Modélisation Mathématique du problème

Mathématiquement le recalage est le processus de recherche de la transformation h, (autrement dit c'est l'estimation du champ de déformation u(x)=h(x)-x) qui vérifier la relation suivante :

$$\text{Optimiser } C\,(Ic(x), Ir\,(h(x) = x + u(x))). \quad \text{Pour tout } x$$

Où : Ω ensemble de toutes les images

C $\Omega * \Omega \rightarrow IR$.

$(X, Y) \rightarrow C(X, Y)$ = taux de similarité entre X et Y.

h : $\Omega \rightarrow \Omega$

$X \rightarrow h(X)$ = image transformée.

Il existe plusieurs classes de recalage, et comme on s'intéresse au cas des images satellitaires les chercheurs proposent de classer ces méthodes selon les critères suivants :

- Les dimensions des images.
- Le type des informations considérées.
- La nature de la transformation considérée et son domaine d'application.
- Interaction machine/homme (approches automatiques, semi automatiques, manuelles).
- La procédure de recherche de la transformation.
- Les modalités (sources d'acquisition).
- La source de données (cartes géographiques, images satellitaires radars ou aériennes).
- L'objet de l'image région d'intérêt, points d'amers …etc.

I.8. PRESERVATION DE TOPOLOGIE

La sortie d'un processus de recalage est une transformation, cette dernière devra corriger une image moyennant une opération de déformation (ou de transformation). La déformation d'une image ne doit en aucun cas modifier les informations globales de l'image d'origine, on parle alors de préservation de topologie.

Préserver la topologie d'une image après une déformation c'est conserver certaines primitives géométriques et certaines caractéristiques. En fait, la topologie est toute une branche de mathématique qui s'intéresse à l'étude de conservation des propriétés géométriques par une déformation, les thèses traitent avec plus de détails cette notion, elles définissent ainsi rigoureusement l'équivalence topologique, et essentiellement les conditions nécessaires pour qu'une déformation préserve la topologie des images. Conserver la topologie est une caractéristique importante que doit garantir une déformation dans certains types de recalage. La déformation doit par exemple maintenir les relations de

voisinages entres structures, ainsi elle doit empêcher l'apparition de nouvelles structures ou la disparition des structures existantes [CHR01].

I.9. TYPE DES PRIMITIVES CONSIDEREES

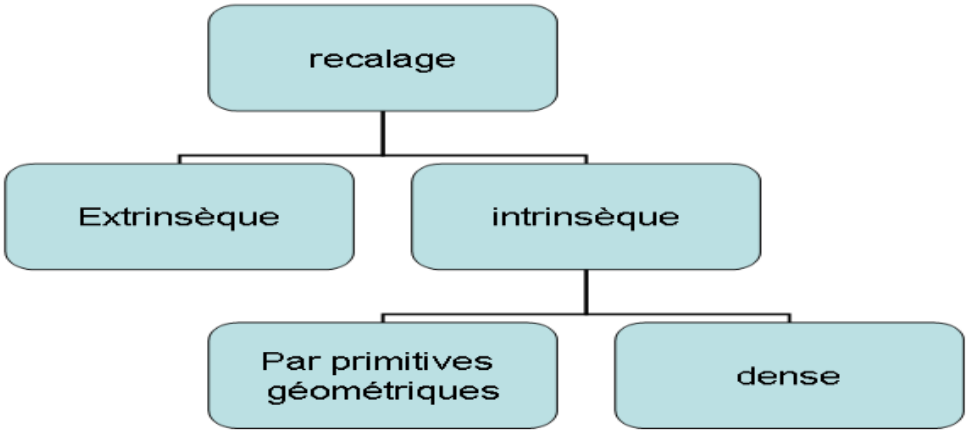


Figure 12. approches possibles pour réaliser un alignement d'images.

Deux grands types d'informations sont considérés, définissant ainsi deux grandes familles d'approches :

I.9.1. Méthodes extrinsèques

Au départ les chercheurs pensaient qu'on peut recaler deux images facilement en mettant au point des dispositifs artificiels dans des emplacements particuliers sur la carte (au moment de l'acquisition). Ces dispositifs laisseront des empreintes remarquables dans les images, le recalage consistera à mettre en correspondance les détailles des dispositifs.

I.9.2. Méthodes intrinsèques

Ce genre de méthodes ne met aucun dispositif lors de l'acquisition des images, le recalage sera basé sur les informations communes des images.

Durant tout notre travail on s'intéresse aux méthodes intrinsèques qui peuvent être divisées en deux grandes approches :

Modèles basés sur des approches par primitives géométriques « **feature_based** » les primitives peuvent être : points, lignes, surfaces ou volumes.

Ceux basés sur des approches denses « **intensity_based** » qui travaillent directement sur les pixels des images.

I.9.2.1. Approches par primitives géométriques «feature_based registration» :

Le principe est le même dans tous les schémas de ce modèle, on s'intéresse à trouver une transformation qui aligne les primitives géométriques fixées préalablement par une méthode généralement semi-automatique, ensuite étendre cette transformation sur le reste de l'image. La transformation optimale doit être cherchée afin de minimiser une distance entre deux ensembles de primitives (par exemple la distance euclidienne).

Ces approches ont beaucoup de problèmes essentiellement : la présence important des minima locaux dans le paysage de recherche ce qui alourdit considérablement le processus d'optimisation, aussi la transformation issue du processus de recherche peut être de bonne qualité, alors qu'elle va perdre encore de qualité après l'étendement sur le reste de l'image.

Une primitive géométrique peut être :

Un point ou Landmark : on commence par la construction de deux ensembles de points (le prélèvement des points se fait généralement d'une façon manuelle), le processus de recherche de la transformation optimale (moindre des carrées) essaie d'aligner les points des deux ensembles. La limite de ce modèle réside essentiellement dans la nécessité d'une intervention manuelle lors de l'extraction des points.

Lignes de crêtes et rubans de surfaces : leurs motivation est la recherche des primitives d'ordre supérieur pour automatiser l'extraction des primitives. En effet, il existe plusieurs procédures automatiques ou dans le pire des cas semi automatiques pour l'extraction de ces derniers (segmentation par exemple). L'approche est la même dans le sens minimisation d'une distance entre deux ensembles.

I.9.2.2. Approaches dense « intensity_based registration »:

Ces approches reposent sur une fonction de similarité (appelée énergie) qui mesure la ressemblance entre images en agissant directement sur les pixels des images, les transformations sont appliquées directement sur la totalité de l'image lors du déroulement du processus de recalage. Dans certains cas, et lorsqu'il s'agit d'une transformation non rigide, la déformation (transformation) doit avoir un grand nombre de degrés de libertés pour pouvoir recaler les images, mais l'information qui existe (des deux images) ne suffit pas pour déterminer (d'une façon unique) la déformation optimale ce qui conduit à un problème « mal posé », il nous faut donc des informations supplémentaires pour garantir l'unicité de la déformation optimale. Pour fournir ces informations nous disposons de plusieurs méthodes : par exemple la régularisation, en exploitant une approche baysiènne ou l'approche la plus intéressante les équations aux dérivées partielles cette dernière qui conduit aux modèles [CHR04 ; CHR01]: élastique et Fluide :

I.10. MODALITES ET REGIONS DE RECALAGE

Rappelons que deux images sont dites monomodales si on utilise la même source d'acquisition, multimodale, si on utilisé des sources distinctes.

I.10.1. Recalage monomodal d'une même région

Un recalage monomodal pour une même région (cartographie), reflète le cas de traitement d'images produites par plusieurs acquisitions, ou des acquisitions dynamiques. Ce genre de recalage est envisagé comme étant un prétraitement avant de performer certaines tâches, par exemple :

- corrections radiométriques ;
- corrections atmosphériques ;
- corrections géométriques ;
- …etc.

I.10.2. Recalage multimodal d'une même région

Ce genre de recalage est utilisé pour accomplir une projection ou une fusion des informations de deux images, la fusion d'images la plus utile est la corrélation thématique.

Pour les deux cas (monomodal/multimodal d'une même région) un recalage rigide s'avère très satisfaisant et il a pour but de compenser la position de la région entre deux acquisitions différentes.

N.B : il faut dire que le recalage rigide est satisfaisant si on exclut les déformations causées par les instruments d'acquisition bien entendue, la satisfaction ou la similitude complète (proche de l'image de référence), repose aussi sur l'idée que le globe ne se déplace pas avec une rigidité, certains reliefs sont complexes à cause de la déformation de la terre (les ruines, les rivières et les cotes maritimes) et nécessitent ainsi un recalage déformable (non rigide) pour régler les imperfections de l'image.

I.10.3. Recalages monomodal/multimodal inter-régions

Recalage monomodal ou multimodal inter-régions (plusieurs cartes terrestres), encore plus complexe et sert à créer des atlas géographiques ou des cartographies opérationnelles (nécessitant dans les plus part des cas des transformations non rigide).

I.11. ELASTICITE DE LA TRANSFORMATION

Les transformations sont généralement classées dans quatre classes selon les effets qu'elles produisent sur les propriétés géométriques d'une image : ***rigide, affine, projective et déformables***.

Une transformation peut être globale ou locale, une transformation globale s'applique sur toute l'image, tandis qu'une transformation locale est un ensemble de transformations appliquées sur des sous parties de l'image en question. Les transformations locales sont presque absentes dans la littérature.

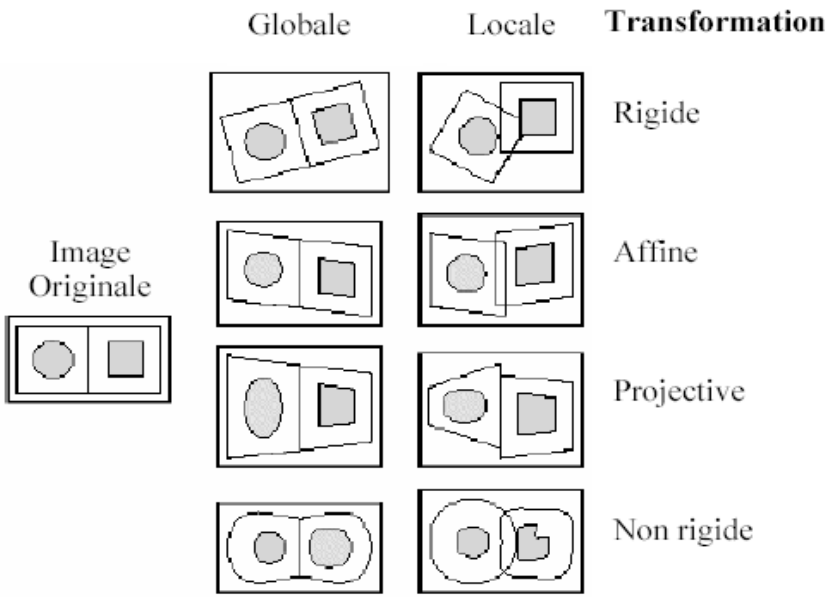


Figure 13. Les effets des différentes transformations.

La transformation rigide n'autorise que des translations et des rotations, alors que la transformation affine conserve l'alignement et le parallélisme. Pour ce qui est des transformations projectives, elles permettent la mise en correspondance de lignes non parallèles entre elles et ne conserve l'alignement que pour les droites horizontales ou verticales. Enfin, une transformation déformable permet de transformer des lignes droites en courbes. La figure 13 illustre les diverses transformations en dimension deux. À noter que l'on trouve d'autres classifications des transformations, notamment en deux classes : transformations rigides; transformations non rigides ou élastiques.

I.11.1. Transformation rigide

Consiste en une rotation et une translation, les distances entre les pixels sont maintenues, les propriétés géométriques (l'alignement et le parallélisme) sont aussi conservées.

La description naturelle d'une transformation rigide est une matrice constante, et le problème revient à trouver un nouvel axe y à partir de l'ancien axe x moyennant la translation t et la rotation r de l'angle α :

$$2D \rightarrow y = r.x + t \Leftrightarrow \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} x1 \\ x2 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}.$$

Equation 2. Formule d'une transformation rigide.

Donc le nombre de variables d'une transformation rigide à estimer est de 3 (deux pour la translation et une pour la rotation) dans le cas 2D.

I.11.2. Transformation affine

Conserve le parallélisme et l'alignement, mais les distances entre pixels ne sont pas maintenues.

En ce qui concerne la description de la transformation affine, on peut garder les mêmes équations linéaires mais sans imposer des restrictions sur la matrice r, les scalaires de la matrice r peuvent prendre n'importes qu'elles valeurs acceptables.

$$2D \rightarrow y = r.x + t \Leftrightarrow \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}.$$

Equation 3. Formule d'une transformation affine.

Et le nombre de variables est 6 (en 2D).

I.11.3. Transformation « projective »

Ne conserve que *l'alignement*.

I.11.4. Transformation déformable (ou non rigide)

Où la transformation est libre, elle ne conserve aucune des trois propriétés mentionnées (Les effets des différentes transformations).

La description d'une transformation déformable peut se faire par n'importe quelle fonction sans restrictions, la façon la plus simple est la représentation linéaire où les nouvelles coordonnées se calculent à partir d'une libre combinaison linéaire des anciennes coordonnées.

I.12. CRITERES DE SIMILARITE

Plusieurs critères de similarité ont été proposés, le choix d'un critère dépend étroitement de l'approche considérée (dense ou géométrique).

I.12.1. Critères de similarité pour une approche par primitives géométriques

Dans le cas d'un recalage basé sur une approche par primitives géométriques, on utilise généralement des fonctions exprimant la distance entre deux ensembles de primitives géométriques par exemple la distance euclidienne, Ces primitives nécessitent un prétraitement à faire (prélèvement des primitives communes). Ce genre de similarité peut être très précieux dans le cas de recalage d'images de plusieurs modalités.

I.12.2. Critères de similarité pour une approche dense

Dans le cas d'un recalage dense on peut envisager plusieurs quantificateurs de similarités reposent sur des formules statistiques mesurant l'interaction entre deux évènements tel que: le rapport de corrélation, la covariance, l'entropie ou encore l'information mutuelle [MAE04] pour plus de critères de similarités, ou encore le critère le plus classique, l'erreur quadratique moyenne :

I.12.3. Erreur quadratique:

$$C(I,J) = \int \left|I(s) - J(s)\right|^2 ds = \sum_{s} \left|\mathrm{I(s)} - \mathrm{J(s)}\right|^2 \quad \text{pour tout pixel s}$$

Equation 4. Formule de l'erreur quadratique.

L'erreur quadratique est toujours positive, et d'autant elle tend vers zéro autant les deux images se ressemblent.

I.12.4. Entropie:

Opère sur une seule image ***D*** qui représente la différence entre les deux images ***I, J :***

$$D = I - J \ .$$

$$C(I,J) = H(D) = -\sum_{s} p(s)\log(p(s)). \quad \textit{pour tout} \text{ pixel s.}$$

Equation 5. Formule de l'entropie

La fonction p décrit la probabilité qu'un pixel ait le niveau de gris s dans l'image D.

L'entropie vient de la thermodynamique, elle reflète le taux d désordre d'une masse physique. Elle est toujours positif, et plus elle tend vers zéro le corps est stable.

I.12.5. Covariance :

$$C(I,J) = COV(I,J) = \sum_s (I(s) - \bar{I}).(J(s) - \bar{J}) \text{ pour tout pixel s.}$$

$\bar{I}$ (resp. $\bar{J}$) : les moyennes des niveaux de gris des images I (resp. J).

Equation 6. Formule de la covariance.

Agissant pixel par pixel, la covariance mesure une corrélation de bas niveau entre les deux images. Une covariance nulle reflète une indépendance totale des deux images (pas de similarité), à l'inverse la similarité est maximale dans le cas où la covariance est maximale en valeur absolue.

I.12.6. Rapport de corrélation :

$$C(I,J) = R^2(I,J) = \left(\frac{COV(I,J)}{\sqrt{\sum_s (I(s) - \bar{I})^2} \times \sqrt{\sum_s (J(s) - \bar{J})^2}} \right)^2 \quad \text{pour tout pixel s.}$$

Equation 7. Formule du rapport de corrélation.

On peut dire que le rapport de corrélation est l'image standardisée de la covariance, car cette dernière varie dans IR alors que le premier varie dans [0, 1].

Si le rapport de corrélation est nul alors il s'agit d'une totale indépendance des deux images, par contre s'il tend vers 1 alors on peut trouver une combinaison linéaire entre les deux images I=a.J+b. à ce stade il faut tester encore les hypothèses statistiques a=1 et b=0 pour confirmer la ressemblance parfaite entre les deux images.

I.12.7. Information mutuelle :

$$C(I,J) = -\sum_{x,y} p(x,y) \log \frac{p(x,y)}{g(x).h(y)}.$$

pour toute combinaiso n de niveaux de gris (x, y).

Equation 8. Formule de l'information mutuelle

p(x, y) est la probabilité (conjointe) qu'un pixel ait le niveau de gris x dans l'image I et y dans J. la probabilité marginale g(x) (resp. h(y)) est la probabilité qu'un pixel ait le niveau de gris x (resp. y) dans l'image I (resp. J).

L'information mutuelle n'assure pas une combinaison linéaire entre les deux images, mais garantie la co-occurrence de maximum de pixels semblables dans les deux images. Ce qui peut être utile dans le cas de recalage multimodal. L'information mutuelle est maximale en cas de ressemblance d'images, elle est toujours positive.

Entropie mutuelle :

$$C(I, J) = -\sum_{s} p(x, y) \log p(x, y).$$

pour toute combinaiso n de niveaux de gris (x, y).

Equation 9. Formule de l'entropie mutuelle.

P(x, y) a la même désignation que celle de l'information mutuelle. L'entropie mutuelle est tout le temps positif, minimal en cas de similarité parfaite.

I.13. CRITERE DE SIMILARITE POUR UN PROCESSUS DE RECALAGE

Les optimums locaux dans le paysage de la fonction de similarité posent le problème le plus crucial lors de la recherche de la transformation optimale. Un processus d'optimisation de recalage, basé sur une fonction qui estime la ressemblance entre images et qui possède beaucoup d'optima locaux, a une forte probabilité de divergence (s'il est stochastique) ou soit de coût calculatoire très élevé s'il est déterministe.

I.13.1. Recherche de la transformation optimale

A ce stade, on peut distinguer deux grandes approches : calculer la transformation optimale, ou la chercher.

Certains modèles peuvent fournir des formules pour calculer la transformation optimale (les modèles élastique et fluide) donc on peut les utiliser pour trouver directement la transformation optimale. Cependant, cette approche est coûteuse en temps de calcul, ce qui nécessite la recherche d'une autre approche « chercher la transformation » moyennant un processus d'optimisation [SAL01].

Un processus d'optimisation de la fonction de transformation T, doit être global pour remédier aux éventuels optima locaux dans le paysage de la fonction de similarité.

Les techniques de recherche de la transformation optimale sont nombreuses, ils sont étroitement liées à la complexité de recalage par exemple pour un recalage rigide basé sur un simple quantificateur de ressemblance on peut utiliser une méthode déterministe (simplex par exemple), mais généralement ces méthodes sont sensibles aux minima locaux ce qui induit un certain retard de convergence qui peut entraîner un surcoût, non négligeable, en temps de calcul machine.

Vu leurs stratégies intelligentes de recherche, L'utilisation des techniques d'optimisation combinatoire basé sur métaheuristiques (recuit simuler, [SAL01] ou algorithmes génétiques [ROU98, SAL01] par exemple) peut réduire le temps de convergence vers une bonne transformation.

De limiter l'espace de recherche en bornant les grandeurs des erreurs en utilisant pour accélérer considérablement le processus de recalage, mais cela diminuera l'efficacité de recalage, et éliminera toute possibilité d'automatisation.

I.13.2. Applications de la mise en correspondance

Les applications du recalage dans la discipline de la télédétection sont très variées citons les plus importantes :

Création des référentiels et des modèles des reliefs : à l'aide d'un recalage monomodal inter-régions on peut construire un modèle présentant toutes les informations communes (mosaïqué les images).

Localisation : des fois, on ne peut pas localiser un relief ou des régions directement vu une éventuelle ambiguïté. Avec une transmission d'informations à partir d'une base de connaissance d'un atlas à travers une déformation vers l'image afin de localiser les structures « floues ».

Fusion des données : la superposition des données pour faciliter l'interprétation, les cartes opérationnelles par exemple.

Correction de mouvements dans une séquence dynamique : afin de remédier à des déstabilisations de la carte lors de l'acquisition de la séquence.

Comparaison entre images. ;

Segmentation et classification d'images ;

…etc.

I.13.3. Formulation – Terminologie

I.13.3.1. Passer de la maximisation à la minimisation

$$\max\left\{C(x)\,\middle|\, x\in\Omega\right\} = -\min\left\{-C(x)\,\middle|\, x\in\Omega\right\}$$

I.13.3.2. Minimum local

- **Configuration x' vérifiant :**

$$\forall\, x\in V(x') \text{ on a } C(x')\le C(x)$$

- **Voisinage de taille ε**

$$V(x') = \left\{x\in\Omega \;\middle|\; \|x-x'\|<\varepsilon\right\}$$

I.13.3.3. Minimum global

- **C'est un minimum local de coût minimum**

I.14. ETAT DE L'ART DU RECALGE

I.14.1. Recalage d'images satellitaires.

Dans le travail entrepris par l'auteur [OMR04], une nouvelle approche basée sur les fonctions de WALSH a été proposée. L'effet du recalage sur cette méthode a été utilisé afin de proposer une mesure qui sera utilisée pour estimer le recalage translationnel entre deux images décalées. L'effet du décalage est plus apparent dans les contours.

En effet basée sur le calcule des cœfficients « F, W » l'approche est basée sur un principe de calcul du dernier pixel en avançant vers le premier pixel. Il est évident que la moindre erreur sera propagée le long du processus de reconstruction. Le vecteur Xn représente l'information manquante dans ce processus de reconstruction. Affecté par les erreurs d'estimation du décalage, ce vecteur sera moins affecté proche de bon décalage entre ces deux images. Dans ce cas, la mesure proposer : elle est donnée par la moyenne de Xn.

Mesure = Mean (Xn).

Le Minimum de cette fonction sera l'estimation du décalage.

I.14.2. Recalage d'images non rigide.

On dispose de deux images, A et B, que l'on voudrait recaler, c'est à dire déformer de façon à ce qu'elles se superposent convenablement. Pour cela, on cherche la déformation h qui, appliquée à l'image A, la fait ressembler a‘ B.

Modélisation

Les images sont des fonctions de IR2 dans IR, ou, plus exactement, si l'on se restreint à un domaine rectangulaire Ω du plan, des fonctions de Ω dans IR.

Nous imposerons (choix arbitraire) que lors du recalage, le bord de l'image reste fixe. Appliquer une déformation h à une image A signifie considérer l'image composée (A ◦ h) ; une déformation h est donc une fonction de Ω dans Ω, valant l'identité sur le bord de l'image @..

Que signifie, pour deux images (recelées) données B et C, qu'elles se ressemblent ? On peut choisir de définir un critère de similarité S(B, C) entre ces images, par exemple

$$S(B,C) = \|B - C\|^2_{L^2(\Omega,\mathbb{R})} = \int_\Omega (B(\vec{x}) - C(\vec{x}))^2 d\vec{x}$$

Equation 10. Critère de similarité.

Qui sera d'autant plus faible que les intensités pixel par pixel des deux images seront proches.

Quel type de déformation s'autorise-t-on ? On souhaiterait obtenir une déformation assez lisse, la plus régulière et la plus faible possible. On peut définir un critère de régularité pour une déformation h :

$$\mathcal{R}(h) = \|h - Id_\Omega\|^2_{H^1(\Omega,\mathbb{R}^2)} = \int_\Omega \|h(\vec{x}) - \vec{x}\|_2^2 + \|Dh(\vec{x}) - Id_{2\times 2}\|_2^2 \; d\vec{x}$$

Equation 11. Critère de régularité.

où IdΩ est la fonction identité de Ω, qui à $\vec{x}$ associe $\vec{x}$, ce qui correspond à la déformation la plus « faible » possible; et où Id2×2 est la matrice identité 2 × 2.

Ainsi, on cherche la fonction h qui minimise le critère E (A, B, h) = S (A ◦ h, B) +R (h).

En fait, on ajoute des poids (réels positifs fixes) devant chacun des trois termes composant le critère à minimiser, afin de pouvoir modifier l'importance relative de chacun de ces termes.

$$\partial_t h = \alpha\,(Id - h) + \beta\,\Delta h + (B - A \circ h)\;\;((\nabla A) \circ h)$$

I.14.3. Méthodes de recalage et de calibrage pour l'aide à l'interprétation d'images médicales en vraies couleurs

Modèles de recalage d'images en vraies couleurs

Appelons CR(i,j) et DR(i,j), le pixel situé à l'intersection des ligne et colonne i et j pour le plan rouge R des images C et D, D étant à recaler par rapport à C. CV(i,j), DV(i,j), CB(i,j) et DB(i,j) sont définis de manière identique pour les plans vert V et bleu B.

Le recalage géométrique

Les images en vraies couleurs sont toujours numérisées dans le système RVB de telle sorte qu'une image est constituée des trois plans Rouge, Vert et Bleu. La partie géométrique du modèle de recalage est identique pour chaque plan de telle sorte que le recalage purement géométrique est identique à celui déjà rencontré dans le domaine de la photographie en noir et blanc [6]. On peut l'effectuer séparément par rapport au recalage couleur en utilisant soit un seul des trois plans soit une combinaison de ceux-ci. Pour des images photographiques, un modèle de recalage géométrique convenable consiste en le produit d'une translation bidimensionnelle T de paramètres x et y, par une rotation G d'angle a et par une homothétie H de rapport h. Ainsi par exemple pour le plan rouge, cette transformation modifie une image D(i,j) en une image D'(i,j) telle que:

$$D'(i,j) = Tx,y * Ga * Hh\,[D(i,j)]$$

Les paramètres x, y, a, h sont communs aux trois plans R, V et B. Le recalage géométrique consiste à trouver les valeurs de x, y, a et h qui rendent l'image D'(i,j) la plus semblable possible à C(i,j).

Le recalage des couleurs

On peut distinguer deux approches dénommées ici globale et composite.

Le modèle linéaire global :

Dans ce cas, le modèle de recalage des couleurs consiste en une transformation linéaire à deux paramètres appliquée identiquement à chaque pixel de l'image, les paramètres étant estimés séparément pour chaque plan RVB.

Ce modèle transforme une image D(i,j) en une image D'(i,j) de telle sorte que:

$$D'R(ij) = \alpha R\, DR(i,j) + \beta R$$
$$D'v(ij) = \alpha V\, DV(i,j) + \beta V$$
$$D'B(ij) = \alpha B\, DB(i,j) + \beta B$$

La procédure de recalage consiste alors à estimer les six paramètres, en plus des quatre paramètres de recalage géométrique, soit au total dix paramètres de recalage, de façon à ce que D'(i,j) soit le plus semblable possible à C(i,j).

Le modèle linéaire composite :

Lorsqu'on applique les mêmes coefficients à un plan donné R ou V ou B, on traite toutes les couleurs de la même manière. Mais physiquement, tous les pigments n'absorbent pas la lumière d'une façon identique, ce qui conduit à effectuer des corrections couleur par couleur. Ceci est réalisé en utilisant d'abord une "quantitative" par une méthode statistique appliquée à la fréquence des couleurs. On extrait de l'image un nombre limité de couleurs principales (par exemple 32) qui résument au mieux l'image de départ. Puis les trois plans RVB sont remplacés par une seule image où figurent les zones correspondant aux couleurs principales. On fait l'hypothèse qu'à chaque couleur principale ainsi déterminée correspond un même comportement de pigment et pour chacune de ces couleurs on calcule une correction suivant un modèle linéaire à deux paramètres. Ainsi, pour les pixels d'une couleur principale donnée (repérés sur l'image composite), pour chaque plan on doit estimer comme précédemment ces paramètres, soit au total 6 * 32 = 192 paramètres de correction de couleur pour l'image considérée, si on en a extrait 32 couleurs principales.

La similitude entre les images doit être quantifiée d'une manière qui rend encore possible le recalage malgré la présence de différences dues à l'évolution des lésions photographiées séquentiellement. C'est la raison pour laquelle doit être construit un critère de similitude entre images couleurs adapté au recalage d'images non similaires.

Exemples d'application

Recalage des couleurs, global et composite

Pour illustrer le recalage des couleurs nous présentons deux images similaires, dans la même situation géométrique. Les couleurs de l'image à recaler sont très fortement déviées vers le rouge. La distance séparant l'image à recaler de l'image de référence (Fig. 1-b), est mesurée par la "Root-mean-square":

$$RMS = \sqrt{\frac{\sum[(C_R(i,j)-D_R(i,j))^2 + (C_V(i,j)-D_V(i,j))^2 + (C_B(i,j)-D_B(i,j))^2]}{N}}$$

Equation 12. Erreur quadratique moyenne.

Elle vaut ici 38.00.

(a) Global: les 6 paramètres sont estimés séparément pour chaque plan RVB.

Après ce recalage global la RMS vaut 23.79.

(b) Composite: l'image de référence est "quantifiée" en un nombre de couleurs assez petit, pour que chaque couleur contienne un nombre suffisant de pixels. 192 paramètres sont calculés pour 32 couleurs principales extraites de l'image de référence. La valeur de la RMS descend à 13.05 avec cette méthode.

PARTIE II. METHODES D'OPTIMISATION POUR LE RECALAGE.

Méthodes d'Optimisation discrète et continue et les Métaheuristiques :

La notion d'optimisation n'est pas récente, les commerçants n'ont pas attendu la programmation linéaire pour essayer de calculer les paramètres optimisant leur gain financier, mais la formalisation mathématique et la création de méthodes de recherche complexes a bien moins d'un siècle. On définit généralement une situation d'optimisation comme le problème consistant à trouver les paramètres à donner à une fonction telle que la valeur retournée par celle-ci soit optimale (minimale ou maximale). On nomme alors fonction objective, ou fonction d'évaluation, la fonction en question. Des contraintes peuvent être imposées pour la recherche de la meilleure solution (espace de recherche, etc.). On peut diviser l'optimisation en deux domaines, selon que le problème soit discret (optimisation combinatoire) ou continu. On pourra aussi trouver des problèmes mixtes, où certaines variables seront discrètes, d'autres continues. La fonction d'évaluation est souvent associée à un problème difficile, dont la solution ne pourrait être trouvée systématiquement avec un algorithme donné en temps raisonnable (problèmes NP-difficiles). Il existe quantité de méthodes pour résoudre ces problèmes, en fonction du besoin (précision, rapidité de calcul, etc.), citons par exemple les méthodes d'approximation réduisant la complexité du problème, la programmation linéaire, ou non-linéaire, les heuristiques spécialisées, ou encore les métaheuristiques, qui sont des méthodes approchées de recherche globale.

Introduction aux métaheuristiques

Un problème d'optimisation est un problème qui se caractérise par une fonction de coût (ou d'énergie) avec ou sans contraintes, défini sur un ensemble fini ou dénombrable de configurations, sur lequel on veut trouver une configuration qui minimise ou maximise la fonction de coût, tout en satisfaisant les contraintes liées au problème.

Lorsque le problème est formalisé avec un ensemble de contraintes, et que le domaine de définition de sa fonction de coût est très grand [YOU98] (on ne peut l'énumérer en un temps raisonnable), alors le problème est dit «problème d'optimisation combinatoire», parmi ces derniers on trouve la classe des problèmes les plus difficiles dits NP-Difficiles.

Résoudre un problème NP- Difficile peut être vu comme, un parcours d'arbre dont le nombre de nœuds (un nœud correspond à faire un choix pour une variable de la configuration) croit exponentiellement suivant sa hauteur qui est polynomiale.

Pour des problèmes NP- Difficile d'optimisation (tel le problème du voyageur de commerce noté par la suite PVC ou le problème de sac multidimensionnel), on ne connaît pas d'algorithme polynomial permettant de les résoudre de façon optimale, on va donc chercher une solution approchée de cet optimum en utilisant des heuristiques.

Les premières heuristiques datent des années 80, et bien que d'origine discrètes, on peut les adapter à des problèmes continus. Elles interviennent généralement quand les méthodes classiques ont échoué, et sont d'une efficacité relativement imprévisible.

Le terme métaheuristique est utilisé car, par opposition aux heuristiques particulières pour un problème donné, les métaheuristiques peuvent être utilisées pour plusieurs types de problèmes, et l'heuristique n'est pas réellement explicite, mais déduite d'un algorithme donné.

Les métaheuristiques ont également comme caractéristiques communes leur caractère plus ou moins stochastique, ainsi que leur inspiration par une analogie avec d'autres sciences (physique, biologie, etc.). On ne peut pas vraiment citer un domaine d'application de prédilection, on pourra retrouver un problème d'optimisation dans de nombreuses situations, l'important pour nous n'étant pas la signification des symboles utilisés (l'application concrète du problème) mais plutôt la complexité de leurs relations. On pourra par exemple livrer une optimisation pour un problème de trafic aérien, de télécommunication (routage), ou pour un classique problème de tournée, et on constatera parfois que deux problèmes sont presque équivalents formellement alors qu'ils ont des domaines d'application totalement étrangers. De toute façon, si deux problèmes difficiles sont NP-Complet, on pourra théoriquement appliquer la résolution de l'un à l'autre dans un temps raisonnable.

Les métaheuristiques ne sont pas des méthodes figées, il n'y a pas de relation d'ordre quant à l'efficacité absolue d'un algorithme ou d'un autre, tout dépend des paramètres utilisés, du problème d'application, d'où l'intérêt de simulations informatisées.

En effet, de nombreuse classifications ont été proposées sur les méthodes (ou algorithmes) de résolutions méthodes déterministes/stochastiques; méthodes de recherche locale/globale ; ou encore méthodes exactes/approchées, on s'intéresse aux classes suivantes

Les méthodes d'optimisation locale

Ces méthodes sont sensibles aux optimums locaux, et elles ne permettent pas d'aboutir à une solution proche de l'optimale globale à cause du grand nombre d'optimums locaux, l'algorithme du gradient (la descente amélioratrice « steepest descent ») est l'un de ces techniques, il est condamné à converger vers une solution qui peut être très loin de l'optimale.

Les méthodes approchées

Ces méthode, elles remédient au problème précédent, elles explorent l'espace de recherche tout en évitant de rester bloqué dans des optimums locaux, ces algorithmes sont de divers variétés comprenant des techniques stochastiques inspirées des phénomènes naturelles, très populaires dans la communauté scientifique (EX Algorithmes Génétiques, Recuit Simulé, etc.). C'est ainsi, que la résolution des problèmes d'optimisation combinatoire, utilise les méthodes approchées, puisque elles évitent les pièges des optimums locaux, et donnent des solutions de très bonne qualité en un temps raisonnable, avec peux d'exigence sur les caractéristiques de la fonction objective comme la continuité et la différentialiste.

En particulier dans ce qui suit, en s'intéresse à ces deux méthodes : l'algorithme génétiques et le recuit simulé qui sont des méthodes de résolution globale approchée.

CHAPITRE II. ALGORITHMES GENETIQUES

II.1. ORIGINE DES ALGORITHMES GENETIQUES

Les algorithmes génétiques (AGis) font partie de la famille des algorithmes évolutionnaires, sont des méthodes adaptatives très utilisées pour résoudre des problèmes d'optimisation. Leurs principes sont basés sur la théorie Darwinienne de l'évolution, qui modélise l'évolution génétique des organismes biologiques, ces derniers vivent ensemble dans un environnement et partagent les mêmes conditions de vie. Dans cette environnement les éléments les plus fort « qui s'adaptent mieux » vont survivre, et ils ont plus de chance de trouver des partenaires pour se reproduire, ainsi la population va se développée au fur et à mesure vers une population plus adaptée à vivre dans l'environnement en question car les éléments les plus forts vont transmettre leurs caractéristiques, à travers les gènes, à leurs descendants. Cette combinaison des meilleurs éléments va donner la possibilité d'avoir des descendants encore plus adaptés à l'environnement encore que leurs parents.

Les algorithmes génétiques ont été inventés aux états unis par J. Holland, en suite largement étudiés par J. De Jong et J.J. Goldberg qui font référence dans ce sens.

Les algorithmes génétiques ont marqué une certaine robustesse en terme des résultats très satisfaisants qu'ils les ont menés. Cette réussite n'a pas été démontrée théoriquement (à notre connaissance), sauf dans un cadre très limité. Mais cela n'a pas empêché l'utilisation des AGs durant toute cette période, par contre l'utilisation des algorithmes génétiques s'accroît exponentiellement chaque année dans des cadres très divers.

II.2. DESCRIPTION ET DEFINITIONS:

D'une manière analogue à celle de l'évolution darwinienne, l'application des algorithmes génétiques à un problème d'optimisation fait évoluer une population d'éléments (individus), où chaque individu représente une configuration (solution) possible, vers une population optimisant ce problème. Pour chaque individu (qui représente une solution) on peut mesurer sa qualité (fitness) qui est la valeur de la fonction à optimiser ou une valeur rapprochée. Pour évoluer on utilise un opérateur de « sélection » qui choisira les meilleurs individus de la population actuelle afin de les mettre dans un « bassin génétique », un opérateur de « croisement » et un opérateur de « mutation » ont pour objectif d'engendrer de nouvelles individus à partir des individus sélectionnés du bassin génétique.

Un algorithme génétique repose donc sur plusieurs notions, ces notions définissent l'hiérarchie suivante

Population ensemble d'individus.

Individu représente une solution. A chaque individu correspond un chromosome régissant les caractéristiques de l'individu.

Chromosome ou phénotype structure sauvegardant l'ensemble des caractéristiques d'un individu, ces caractéristiques sont généralement codées. Dans un langage génétique une caractéristique correspond à la notion de gène. Un chromosome dans un problème d'optimisation correspond à l'ensemble des variables à optimiser.

Gène un champ dans un chromosome régissant les caractéristiques d'une partie de l'individu. A un gène correspond un type, un type est en quelque sorte une image codée d'un gène. Un gène est une variable à optimiser dans un problème d'optimisation, et un type est le codage de variable.

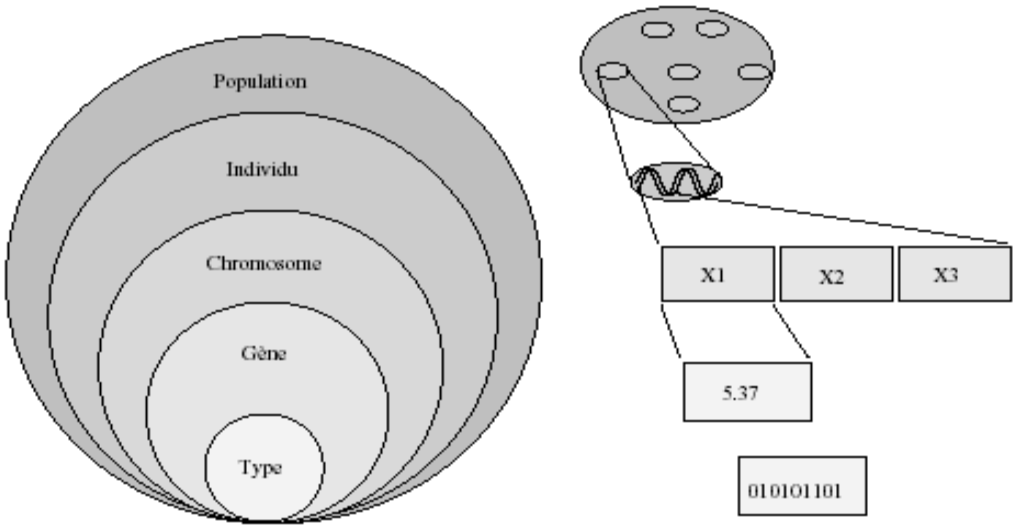


Figure 14.: hiérarchie dans un algorithme génétique.

Et la forme générale d'un algorithme évolutionniste génétique est la suivante

Initialisation ;
Evaluation ;
Répéter
    Evolution ;
    Sélection ;
    Croisement ;
    Mutation ;
    Evaluation ;
Jusqu'à (critère d'arrêt).

Algorithme 1. Forme générale d'un algorithme évolutionniste

Donc les algorithmes génétiques sont des méthodes descendantes, dans le sens où la solution optimale sera trouvée en regroupant les caractéristiques optimales (gènes optimaux) à partir des solutions parents. Si les caractéristiques optimales n'ont pas la chance d'être générées alors l'algorithme ne convergera jamais vers la solution optimale.

La performance et la complexité d'un algorithme génétique dépendent étroitement, aux caractéristiques suivantes

Le codage pour définir l'espace d'intervention des opérateurs de croisement et de mutation. Le codage c'est la définition de la représentation génétique d'une solution, i.e. définir la correspondance entre l'espace des solutions (phénotypes) et l'espace des chromosomes (génotypes).

La taille de la population.
La fitness la qualité d'une solution, régie par une fonction de coût.
Le mode de sélection, et les opérateurs de reproduction.
Et d'autres paramètres (taux de croisement, de mutation,…etc.)

Codage :

Dans ce qui suit on représente une solution de dimension n par $X = \{x_1, x_2, ..., x_n\}$, l'espace phénotypique des solutions soit : Ω.

Le codage le plus répandu dans la littérature est sans doute le codage binaire, c'est le meilleur car il est aveugle du fait qu'il ne voie pas la nature du problème. Dans ce sens un génotype (ou chromosome) est un mot appartenant au langage itéré sur l'alphabet {0, 1}.

Ce genre de codage peut être appliqué à pas mal de problèmes par exemple le problème de voyageur de commerce, ou les villes sont numérotées de 0 à 7. Une solution possible soit

1→5 →6 →0 →3 →4 → 2→ 7→1 un codage binaire possible est la chaîne binaire suivante [001 101 110 000 011 100 010 111 001] = [1 5 6 0 3 4 2 7 1].

D'une façon générale le codage binaire peut être appliqué d'une façon très naturelle aux problèmes traitant des grandeurs discrètes (i.e. $\Omega = Z^n$). Dans le cas de traitement des grandeurs continus on peut procéder à un échantillonnage de l'espace de travail, bien qu'avec cette approche on puisse coder les solutions par des chaînes binaires, l'algorithme ne converge pas vers une solution optimale mais vers une solution approchée. Il est évident que l'écart entre la solution optimale et celle issu de l'algorithme s'affaibli lorsqu'on choisit un pas d'échantillonnage très petit, mais choisir un pas très petit revient à fournir plus d'espace de stockage pour les solutions ce qui prohiber l'approche de codage binaire des grandeurs continus (si on veut de la précision).

Et pour remédier à ce problème, un codage non binaire est envisagé (par exemple le codage quantique dans notre développement), et consiste souvent de travailler directement sur la solution elle même i.e. travailler directement sur la forme phénotypique de l'individu, et dans ce cas on rejoint d'autres algorithmes évolutionnistes les stratégies évolutionnaires ou encore l'évolution différentielle.

II.2.1. La population initiale

Pas mal de travaux de recherche dans le cas des algorithmes génétiques suivant un schéma canonique ont montré la convergence de l'algorithme quelque soit la configuration initiale. Donc on peut choisir manuellement une population de départs, ou bien on peut utiliser un générateur aléatoire de solutions, cette approche garantie en quelque sorte la diversité de la population initiale ce qui conduit à une meilleure exploration de l'espace de recherche.

II.2.2. Mesurer la qualité (fonction de fitness)

Tout d'abord il faut décoder le génotype, de trouver l'image phénotypique du chromosome, en suite calculer sa qualité suivant la fonction F. Cette dernière peut donner une information exacte ou approchée de la qualité de la solution, elle doit être bien conçue afin de garantir le bon fonctionnement et la convergence de l'algorithme.

Au fur et à mesure la progression de l'algorithme les individus vont se ressemblés dans le sens de leurs fitness à pleine échelle. La sélection ne peut se faire facilement dans ce cas, et il va falloir changer d'échelle afin de rendre l'opérateur de sélection plus évident. Le changement d'échelle le plus utilisé est l'ajustement par échelle linéaire $F'=b+a.F$. Avec a, b sont des réelles et a non nul.

Exemple les deux nombres 10,509 et 10,501 sont pratiquement égaux (précision réduite à 1/100) à pleine échelle, si deux individus aient des qualités semblables à ces deux grandeurs alors il est difficile de sélectionner un, alors que si on ajuste linéairement ces deux grandeurs en les multipliant par 100 on obtient 1050.9 et 1050.1 qui sont différents (précision 1/100), donc on peut facilement sélectionner l'individu dont la fitness est 1050.9.

II.2.3. Les opérateurs

II.2.3.1. La sélection

Cette étape a pour but le choix des meilleurs individus pour la reproduction, ce choix là peut être déterministe (ou élitiste), les meilleurs individus sont choisis suivant leurs qualités les autres vont disparaître. Dans ce cas la diversité n'est pas garantie donc l'exploration de l'espace de recherche est mauvaise. Ce schéma conduira sans doute à un optimum local suivant une convergence prématurée.

La sélection aléatoire peut remédier à ce problème, mais certainement les meilleurs individus doivent avoir des grandes chances d'être retenues. D'une façon générale un individu x_i a la probabilité $P_s(x_i)$ d'être sélectionné, bien entendu cette probabilité dépend d'une façon ou d'une autre de la qualité de l'individu. Dans la littérature on trouve plusieurs approches pour calculer cette probabilité citons les plus répandue

1. Roue de la fortune (ou de loterie)

Avec une population de M individus nommés $x_1,...x_M$ la probabilité de sélection est calculée de la façon suivante

$$i \in \{1,.., M\}, p_s(x_i) = \frac{F(x_i)}{\sum_{j=1}^{M} F(x_j)}.$$

Equation 13. Probabilité de sélection dans le cas "roue de la fortune".

Chaque individu possède une case dans la roue, l'angle de la case dépend de la fitness de l'individu, ainsi si la fitness de l'individu est relativement grande il va être sélectionné plusieurs fois, sa descendance dominera la population. De cette façon la convergence soit très rapide vers cet individu on suit alors une convergence prématurée.

Ce mode de calcul de la probabilité de sélection, rejoint le principe du tirage aléatoire par la roue d'un casino. Chaque individu de la population aura une entrée dans la roulette, l'espace réservé à chaque entrée sera proportionnel à la fitness de l'individu qui l'occupe.

Individus	*fitness*	*fitness relative*
1	15	0.1
2	30	0.2
3	45	0.3
4	22.5	0.15
5	7.5	0.05
6	30	0.2

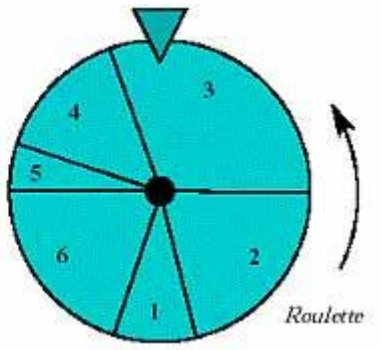


Figure 15. Principe de la sélection par roue de la fortune [MAT02].

Avec cet opérateur, si la fitness d'un individu est relativement grande il va être sélectionné plusieurs fois, sa descendance dominera les populations futures. De cette façon la convergence soit très rapide vers les caractéristiques de cet individu, on suit alors une convergence prématurée.

2. Rang de classement

Est une solution pour remédier au problème de la convergence prématurée de la sélection par rue de fortune, dans cette approche les individus sont triés et classés dans une liste selon leurs fitness. On calcule alors une autre fonction de fitness proportionnelle à leurs rangs dans la liste de la façon suivante (à titre d'exemple)

$$F'(x_i) = MAX - (rang(x_i) - 1) \times \frac{MAX - MIN}{M - 1}$$

Avec **rang(x)=** le rang de l'individu **x** dans la liste.
1<MAX<=2 & MIN = 2-MAX

Equation 14. Fonction de fitness.

On peut par la suite calculer, en basant sur la nouvelle fitness des probabilités de sélection, et d'appliquer un critère de sélection (sélection par roue de fortune par exemple).

Dans ce schéma, les meilleurs individus ont bien entendu plus de chance d'être retenue, un individu de qualité inférieur a une chance d'être retenue lui aussi. Dans ce le croisement crée une diversité ce qui conduit à une meilleure exploration de l'espace de recherche.

3. Sélection par tournoi

Dans un cadre de traitement parallèle, ou si la taille de la population est trop grande, le calcule des probabilités devient coûteux en quelque sorte, l'idée est de minimiser la quantité des informations à traiter. En effectuant un prélèvement d'un échantillon (de taille strict supérieur à 1) d'individus aléatoirement, les individus de cet échantillon entrent alors en compétition les meilleurs individus sont alors choisis.

Les opérateurs de sélection présentés au dessus, ne sont pas les seuls. Une remarque à faire à ce stade là est que la sélection est indépendante du schéma de codage, qu'il soit binaire ou non binaire, on peut bien sélectionner les individus car on n'aura besoin que de la fitness des individus et celle-là repère le niveau supérieur (phénotypique).

II.2.3.2. Le croisement

Le croisement est un opérateur génétique binaire permettant la création de nouveaux individus enfants à partir de leurs individus parents, à l'aide de cet opérateur les parents passent leurs caractéristiques à leurs enfants. D'une façon plus matérialisée le croisement est un échange par sous chaînes (blocs) au niveau des chromosomes, ces sous chaînes sont déterminées par des points de croisements prélevés aléatoirement.

Le croisement se fait (généralement) suivant une probabilité de croisement Pc qui doit être élevés pour garantir l'intervention de l'opérateur, relativement la création de nouveaux individus.

a. Croisement pour codage binaire

Dans la littérature existe deux types de croisement, croisement simple et croisement multiple de niveau k > 2.

Croisement simple dans ce cas il n'existe qu'un seul point de croisement généré souvent aléatoirement. Ce genre de croisement est très utile si la taille d'un chromosome est très petite, cependant il risque d'être improductif dans le cas contraire dans le sens ou il soit incapable de générer de nouveaux individus.

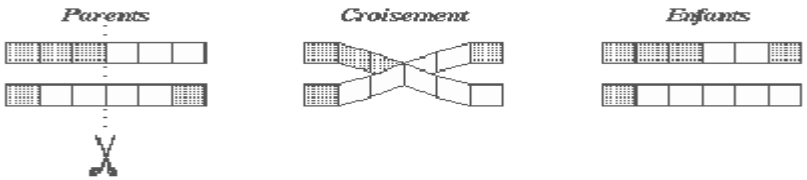


Figure 16. Croisement simple.

Croisement multiple de niveau k>2

Il revient localiser (k-1) point de croisement sur les chromosomes parents, définissant ainsi k blocs (sous chaînes) dans les deux chromosomes. Le schéma le plus général du croisement est que les deux enfants prennent les blocs de leurs parents alternativement. Mais dans la nature on trouve plusieurs approches qui introduisent le caractère stochastique lors de l'échange des blocs. En effet les deux enfants ont la même chance ou des chances différentes d'avoir un bloc particulier de l'un des parents.

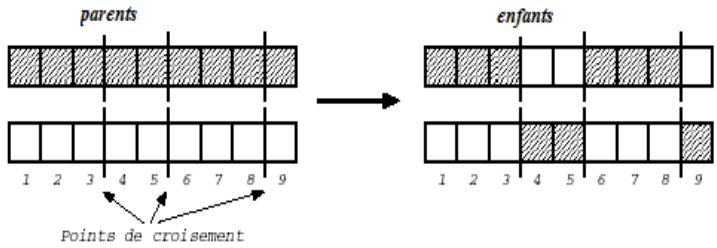


Figure 17. Croisement multiple de niveau 4.

Le croisement multiple est un excellent choix pour diversifier la population (ne pas générer des individus déjà généré), dans le cas ou la taille du chromosome soit important.

Croisements pour codage non binaire réel

Comme nous l'avons vue une solution peut ne pas être codée en une chaînes de booléens, mais d'une façon plus générale elle peut être vue comme un vecteur de n caractéristiques $X = \{x_1, x_2, ..., x_n\}$.

L'approche la plus intuitive, est de voir ces caractéristiques comme des bits et d'appliquer les mêmes opérateurs de croisements définis pour le codage binaire. Mais cette approche ne permet pas l'exploration des espaces des caractéristiques. Donc il faut trouver d'autres solutions, par exemple emprunter les opérateurs de reproductions des autres algorithmes évolutionnistes (stratégies d'évolution par exemple)

Supposons qu'on a deux parents P1 et P2 de caractéristiques respectives $\{p_{11}, p_{12}, ..., p_{1n}\}$, $\{p_{21}, p_{22}, ..., p_{2n}\}$

Le croisement arithmétique tiens compte des grandeurs des caractéristiques des parents par combinaison linéaire où les informations des deux parents sont pondérées par des taux complémentaires uniformes dans [0,1].

$$enfant_1 = \alpha.(p_{11},...,p_{1n}) + (1-\alpha)(p_{21},...,p_{2n}).$$
$$enfant_2 = (1-\alpha)(p_{11},...,p_{1n}) + \alpha.(p_{21},...,p_{2n}).$$
$$\alpha \longrightarrow uniforme\_dans\_[0,1].$$

Dans ce schéma toutes les caractéristiques sont pondérées globalement, cependant on peut pondérer chaque paramètre isolément.

$$enfant_1 = (\alpha_1 p_{11} + (1-\alpha_1) p_{21},...,\alpha_n p_{1n} + (1-\alpha_n) p_{2n}).$$
$$enfant_2 = (\alpha_1 p_{21} + (1-\alpha_1) p_{11},...,\alpha_n p_{2n} + (1-\alpha_n) p_{1n}).$$
$$\alpha_i \longrightarrow uniforme\_dans\_[0,1].$$

Le croisement linéaire permet une exploration meilleure de l'espace de recherche.

II.2.3.3. La mutation

La mutation est un opérateur génétique unaire, il se caractérise par la modification, avec une probabilité de mutation Pm, de la structure d'un chromosome. L'approche la plus classique de mutation fondée sur une représentation binaire est d'inverser des bits dans la chaîne d'un chromosome selon la probabilité de mutation. L'opérateur peut provoquer le changement de plusieurs bits.

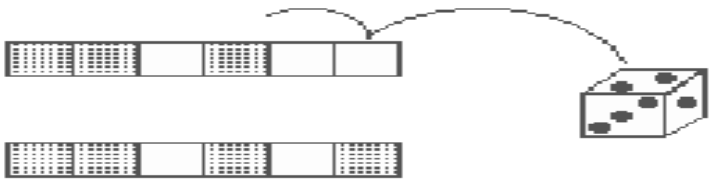


Figure 18. Mutation en codage binaire.

Dans le cas d'un codage réel on peut utiliser la même approche pour changer la valeur d'une caractéristique, par exemple d'ajouter ou de retrancher des valeurs arbitraires.

La mutation permet d'engendrer des nouveaux individus, donc conduit à une meilleure exploration de l'espace de recherche, elle permet aussi d'échapper à un piège d'un optimal local dans un cas de convergence vers ce dernier.

La probabilité de mutation dans un processus d'évolution biologique naturelle est très petite, dans un processus d'optimisation basé sur algorithme génétique le doit être aussi, car la philosophie d'évolution génétique impose que le croisement soit l'opérateur le plus important de l'évolution.

II.2.3.4. Croisement et mutation et leurs fonctions dans les AGs

Il est trivial qu'un bon processus d'optimisation doit explorer d'une manière intelligente l'espace de recherche afin de tomber rapidement sur l'optimum global ce qui

traduit généralement par la recherche dans des nouvelles régions. Et il est aussi évident que ce bon processus doit exploiter les informations déjà traités (des solutions déjà visitées) afin de bien guider la recherche [SUN04].

Pratiquement les processus stochastiques ont montré leurs efficacités d'exploration. Cependant ces processus ne favorisent pas l'exploitation des informations, encore plus qu'on désire exploiter les informations on diminue relativement la qualité d'exploration. Donc le même bon processus doit trouver un bon compromis entre exploration et exploitation.

Un algorithme génétique semble lui aussi être un bon processus d'optimisation dans le sens où les opérateurs de croisement et de mutation assumes tous les deux les fonctionnalités de l'exploration et de l'exploitation, mais avec des grandeurs différents, dans le sens ou le croisement favorise plus l'exploration à l'inverse de la mutation qui favorise elle l'exploitation. Donc il ne nous reste que trouver le bon compromis.

II.2.4. Critère d'arrêt

Naturellement, on ne peut avoir des nouveaux individus (portant des nouvelles caractéristiques) à partir d'une population homogène dont les individus se ressemblent (i.e. portant les mêmes gènes).

D'une manière analogue, on dit qu'un processus d'optimisation basé sur un algorithme génétique a convergé si la population actuelle est homogène. Une population est homogène si la totalité ou la majorité des individus portent les mêmes gènes (cas discret) ou des gènes dont les valeurs se rapprochent (cas continu).

Le jugement de l'homogénéité d'une population peut s'appuyer directement sur les structures phénotypiques des individus, en testant la ressemblance des gènes, mais cela peut entraîner un coût calculatoire important dans le cas des grands problèmes nécessitants un grand nombre de variables à optimiser. D'où la nécessité de trouver un indicateur, moins coûteux, pour témoigner la convergence de l'algorithme.

Des grandeurs statistiques sur l'évolution des fitness des populations peuvent être chaleureusement adaptés, citons par exemple la moyenne des fitness de tous les individus rencontrés (performance en ligne) ou encore la moyenne des fitness des meilleurs individus par génération (performance hors ligne). Ces grandeurs convergentes vers des valeurs stables suivant la convergence de la population, ainsi on peut juger la convergence de l'algorithme par la stabilité de l'un des deux indicateurs cités précédemment.

II.2.5. Réglage des paramètres d'un algorithme génétique

Tout au long de ce texte, on est arrivé à définir plusieurs paramètres, en plus on a imposé même des contraintes sur certains paramètres (probabilité de croisement qui doit être élevée, et probabilité de mutation faible). Dans ce paragraphe on ne revient pas sur tous les paramètres mais sur quelques uns, en montrant l'influence de chacun sur la performance de l'algorithme.

Commençons par la taille des populations, une taille très faible risque de ne pas créer une diversité, ainsi l'exploration de l'espace sera mauvaise et le processus risque de suivre une convergence prématurée. Tandis qu'une taille très grande risque de ralentir le processus d'optimisation à cause du nombre important d'individus à traiter, ce qui ralenti

considérablement la convergence. En ce qui concerne le taux de mutation, on avait dit que naturellement ce dernier doit être relativement faible, en effet avec une probabilité de mutation élevée les individus auront plus de chance d'être mutés, et comme une mutation a un statut aléatoire, la recherche aura le même statut et favorisera ainsi l'exploration mais pas l'exploitation. De même une probabilité très faible risque de conduire le processus d'optimisation à une convergence prématurée, à cause de la faible exploration de l'espace de recherche. Un taux de croisement très faible provoque une mauvaise exploration et une mauvaise exploitation.

II.2.6. Schéma de déroulement de l'algorithme génétique

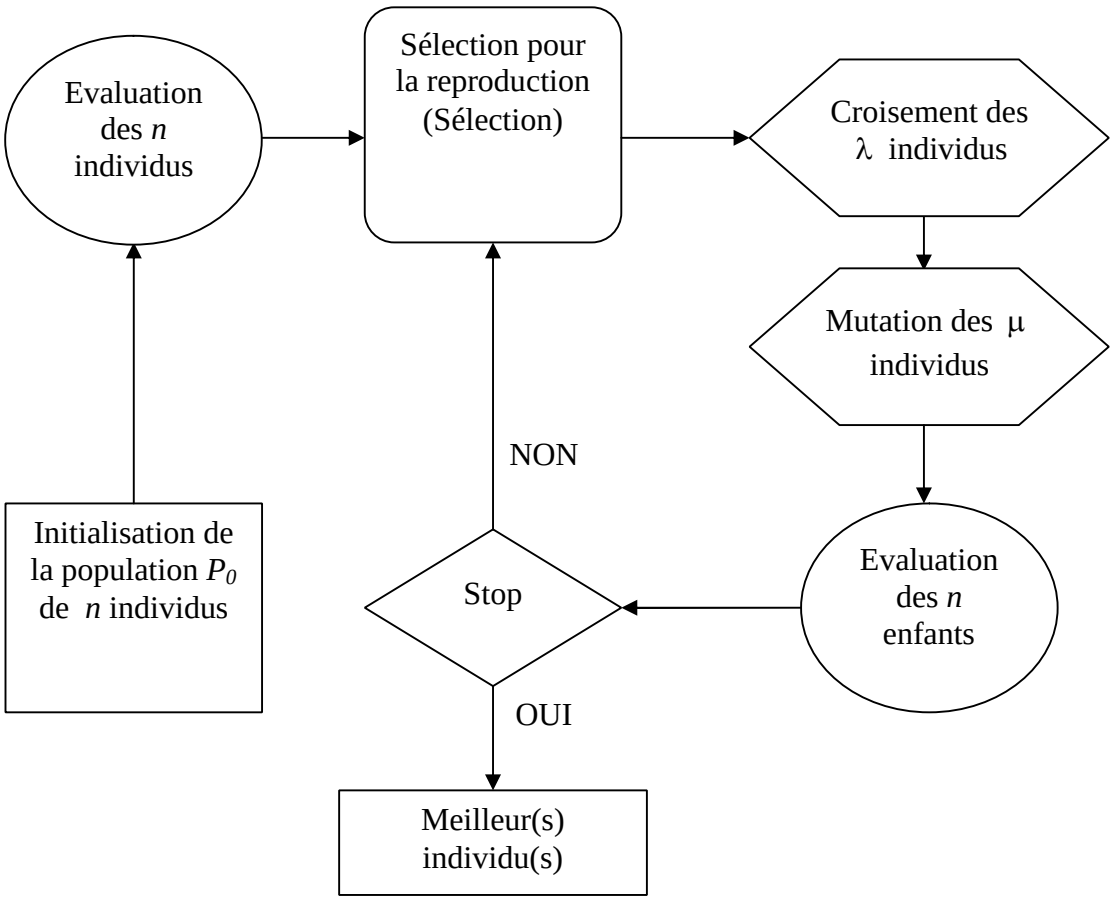


Figure 19. Organigramme de déroulement d'un Algorithme génétique.

CHAPITRE III. RECUIT SIMULE

III.1. ORIGINE DU RECUIT SIMULE

Le recuit simulé est une version améliorée de la méthode d'amélioration itérative. Il a été proposé en 1983 par Kirkpatrick pour la résolution d'un problème de placement en VLSI (Very Large Scale Integration). La méthode inspirée du principe thermodynamique, dans lesquels les déplacements dans l'espace de recherche sont basés sur la distribution de Boltzmann, cette dernière mesure la probabilité de trouver un système dans une configuration X d'énergie E(X), à une certaine température T donnée, dans l'espace de configurations U et elle est définie par :

$$P(X)= \frac{1}{Z_T} \exp\left(- \frac{E(X)}{K_B T} \right)$$

Z_T constante de normalisation ;

Et K_B constante de Boltzmann

Equation 15. Distribution de Boltzmann

Dans cette expression, le facteur $K_B T$ montre que lorsque la température est très élevée, tous les états sont à peu prés équiprobables, c'est-à-dire un grand nombre de configurations sont accessibles, par contre quand la température est basse, les états à hautes énergies deviennent peut probables par rapport à ceux de faibles énergies.

III.2. DESCRIPTION DE L'OPERATION PHYSIQUE « RECUIT »:

En effet, l'état de la matière dépend de la température: à haute température, on a l'état liquide et à basse température, l'état solide.

Si l'on souhaite obtenir un métal avec une structure parfaite de type cristal (c'est l'état fondamental correspondant au minimum d'énergie interne), on va procéder comme suit après avoir porté la matière à l'état liquide, on va abaisser la température jusqu'à solidification. Si la décroissance de température se fait de façon très brusque, on obtient un 'verre', caractéristique de la technique de 'trempe'. Si par contre elle se fait de façon très progressive, laissant le temps aux atomes d'atteindre l'équilibre statistique, on tendra vers des structures de plus en plus régulières pour finir dans l'état fondamental: le cristal, caractérisant le gel du système.

Au cas où cet abaissement de température ne se ferait pas assez lentement, il pourrait apparaître des défauts. Il faut alors les corriger en réchauffant de nouveau légèrement la matière de façon à permettre aux atomes de retrouver la liberté de mouvement, leurs facilitant ainsi un éventuel réarrangement conduisant à une structure plus stable. Ce réchauffement porte le nom de recuit.

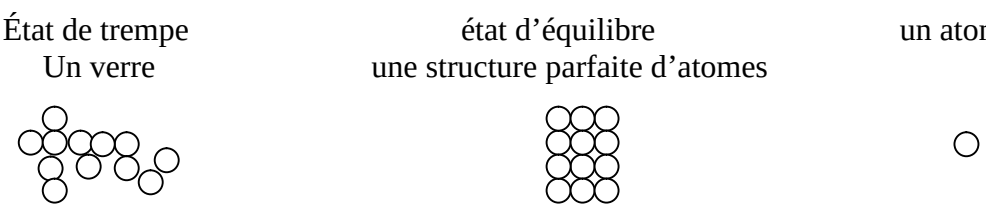


Figure 20. Etats physiques de la matière.

En s'inspirant de principe de recuit qui est appliqué pour chercher des états stables pour les matières, le recuit simulé cherche à trouver des solutions de très bonne qualité pour un problème d'optimisation.

Donc la matière est vue comme étant une fonction de coût du problème à optimiser, et les états de cette matière peuvent ressembler à des solutions de la fonction en question.

A présent, le processus de recherche peut être vu comme suit

Dans un premier temps, on a une température très haute c'est-à-dire que toutes les configurations de l'espace de recherche sont acceptables, et on va baisser la température lentement pour que les configurations acceptables tendent vers la configuration optimale, la recherche s'effectue par laps, à chaque fois qu'on atteint un état quasi-équilibre on baisse la température, ainsi on va modeler la recherche pour quelle aboutisse à une solution proche de l'optimale. En effet, contrairement aux autres méthodes (ex. la méthode du gradient, …), un nouveau trajet de coût supérieur à celui du trajet courant ne sera pas forcément rejeté, son acceptation sera déterminée aléatoirement, en tenant compte de la différence entre les coûts et la température actuelle.

Le concept de température d'un système physique n'a pas d'équivalent direct dans un processus d'optimisation de type recuit simulé. Le paramètre température T est simplement un paramètre de control, indiquant le contexte dans lequel se trouve le système (ex: stade de la recherche). En fait, le paramètre T contrôle les déplacements vers les points voisins les moins bons pour échapper aux optima locaux, sans pour autant trop s'écarter du chemin vers le vrai optimum.

Un critère de voisinage est imposé naturellement, dans un processus de cristallisation du matériel, entre les états. Dans le sens où un laps de refroidissement défini deux états voisins : l'état du matériel avant changement de température, et l'état du matériel après le changement. En gardant l'analogie, les déplacements dans un processus d'optimisation par recuit simulé ne peuvent se faire qu'entre deux solutions voisines.

L'algorithme de Kirkpatrick simule ce processus en combinant dans l'algorithme les mécanismes de refroidissement et de recuit (Figure I.3.1)

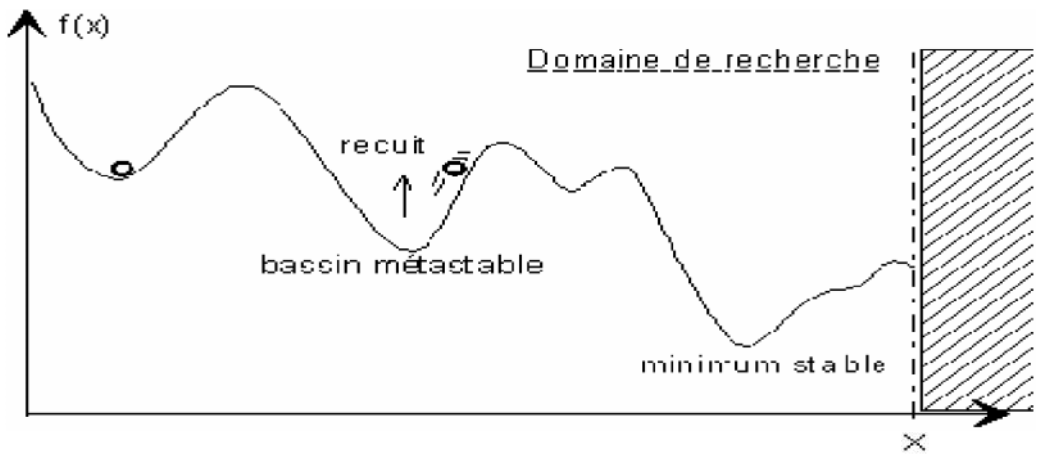


Figure 21. Parcours de l'espace de recherche avec le recuit simulé.

Le principe de "recuit" qui se traduit par une augmentation du niveau d'énergie, permet de sortir des minimums locaux.

En ce qui concerne l'acceptation des nouvelles solutions dans l'algorithme de recuit simulé, la probabilité de Boltzmann n'est pas directement appliquée, mais le critère de Metropolis est utilisé. Le critère de Metropolis permet de décider si une nouvelle configuration générée présente une variation de coût acceptable. Il permet de décider aussi de sortir des minima locaux quand le critère d'arrêt n'est pas encore atteint.

III.2.1. Critère de Metropolis

À chaque passage d'une configuration X à une configuration Y, on calcule la variation de la fonction de coût $\Delta f = f(y) - f(x)$ passage est accepté selon la probabilité p(X, Y) telle que:

$$p(x,y) = e^{\left(\frac{f(y)-f(x)}{T}\right)} = e^{\left(\frac{-\Delta f}{T}\right)}$$

Equation 16. Probabilité d'acceptation selon le critère de Metropolis.

Lorsque la variation $\Delta f \leq 0$, l'exponentielle est supérieure ou égale à 1, la nouvelle configuration doit être acceptée, en lui affecte alors la probabilité maximale de 1. Donc la probabilité p(x, y) se résume Comme suite

$$p(x, y) = \min(1, e^{\frac{-\Delta f}{T}})$$

Si $\Delta f > 0$, on compare p(X, Y) à un nombre aléatoire **R** dans [0,1[:

- Si **R** < p(**X, Y**) la configuration **Y** est acceptée;
- Sinon elle est rejetée et on essaie une autre configuration.

Au début de l'algorithme, le facteur T est élevé, la probabilité p(X, Y) est proche de 1 et presque toutes les configurations de l'espace de recherche sont acceptables. Au contraire, quand T diminue, les remontées vont être de plus en plus difficiles et seules de très faibles déplacements peuvent être acceptées. Si une configuration est rejetée, le système essaie d'en trouver une autre, sinon elle est acceptée et la recherche continue avec celle-ci jusqu'à ce que le critère d'arrêt soit atteint.

III.2.2. Caractéristiques du Recuit Simulé

III.2.2.1. Modélisation Markovienne

Une chaîne de Markov est une suite d'états finis aléatoires, liés entre elles par des probabilités qui constituent une matrice de transitions ; les états sont des configurations visitées à la température T_k. Lorsque T_k est constante, la probabilité est homogène. Et si le nombre de transitions tend vers l'infini, l'état le plus probable apparaît plus souvent et on obtient alors l'équilibre statistique à cette température. Théoriquement, on montre que si l'on fait tendre la longueur de la chaîne de Markov vers l'infini, on peut obtenir la convergence asymptotique de l'algorithme.

En pratique, il n'est pas toujours possible de connaître le nombre d'itérations nécessaire pour atteindre l'optimum global, de plus le temps de calcul est limité, donc la convergence asymptotique ne peut être qu'approchée. A cause de cette approximation, l'algorithme de recuit simulé ne peut garantir d'atteindre l'optimum global avec une probabilité égale à 1. Ainsi, pour augmenter la chance d'obtenir une solution la plus proche de l'optimum global, il est important de trouver les valeurs de compromis entre les paramètres de contrôle de l'algorithme.

III.2.2.2. Voisinage et sélection

Une solution du problème d'optimisation par recuit simulé définie un voisinage. Le voisinage est l'ensemble des solutions qui ressemble (en terme de caractéristiques, et non pas en terme de qualité), donc on dit qu'une solution est la voisine d'une autre si la « morphologie » de la première ressemble à celle de la deuxième (à quelque différences). Pratiquement on « calcul » la solution voisine à partir d'une solution en lui apportant quelques petites modifications. Dans le cas discret ou les solutions peuvent être vues comme étant une chaîne de paramètres, on peut voir la fonction qui calcule le voisinage d'une solution comme étant des petits changements d'un ou plusieurs paramètres. Ce dernier ressemble énormément à l'opérateur de mutation des algorithmes génétiques.

La sélection d'une solution voisine à adoptée est aléatoire. Donc durant le processus d'optimisation on choisit arbitrairement une solution voisine, si celle-là va être adopté selon le un critère particulier alors l'algorithme continuera avec la voisine, sinon on tente notre chance avec une autre solution voisine.

Au fur et à mesure de progression de l'algorithme d'optimisation, on peut revenir à des solutions déjà visitées, cependant la probabilité de revenir devient très faible si l'espace de recherche est trop large, et la fonction qui génère le voisinage est bien conçue.

III.2.2.3. Température

La température est un paramètre de contrôle. Dans le sens où plus la température est haute plus on peut admettre des nouvelles solutions i.e. remédier au problème d'optimum locaux, et plus la température est basse plus on s'oriente vers l'optimum global.

En diminuant lentement la température, la recherche va passer de étape (ou laps) vers étape tout en assurant le quasi-équilibre, et c'est important pour assurer une meilleure exploration de l'espace de recherche, et d'augmenter la chance d'atteindre la solution optimale.

La loi selon laquelle la température décroît est également importante pour l'efficacité de l'algorithme, puisqu'elle doit laisser le temps à l'algorithme de tester le maximum de configurations. De plus, il faut tenir compte de la température initiale qui doit être suffisamment élevée pour que la descente en température soit aussi lente que possible.

1. Température Initiale

La valeur de la température initiale T_0 doit être choisie élevée. Elle est déterminée, sinon fixée arbitrairement, lors d'une phase de prétraitement avec une exploration initiale partielle de l'espace de configurations. T_0 Doit être choisie de sorte que la probabilité d'acceptation de la plus mauvaise solution soit environ 80% (i.e. P(x, y) = 0.8)

2. Décroissance de la température

Le changement de température T_k vers T_{k+1} est déterminé par le moment où l'on a détecté l'équilibre statistique (ou l'état du quasi-équilibre) à la température T_k. La variation de température se fait donc par "palier" suivant la fonction de décroissance utilisée. Les fonctions les plus couramment rencontrées dans la littérature sont les fonctions linéaires, discrètes ou exponentielles (Voir Tableau 3).

Type	**Fonctions**	**Paramètres**
Linéaire	$T_{k+1} = \alpha \times T_k$	$\alpha < 1$ (entre. 0.95 à 0.8)
Discrète	$T_{k+1} = T_k - \Delta T$	$\Delta T > 0$, pour la descente $\Delta T < 0$, pour le recuit
Exponentielle	$T_{k+1} = T_k . exp(\frac{-\lambda . T_k}{\sigma_k})$ avec $0 \leq \frac{T_{k+1}}{T_k} \leq 1$	S_k, l'écart type des coûts des configurations acceptées sous la température T_K , le paramètre de réglage fixé par l'utilisateur

Tableau 3. Les Fonction de décroissance de températures les plus utilisées.

La loi linéaire permet d'avoir une décroissance ni trop rapide (discrète) ni trop lente (exponentielle).

La décroissance exponentielle permet de tenir compte de l'état précédent par l'utilisation de l'écart type des coûts obtenus sous le palier de température précédent. Au début de la recherche, un maximum de configurations est accepté. Comme ces configurations peuvent être très dispersées dans l'espace de recherche, alors l'écart type doit être relativement grand et donc la température décroît très lentement. La décroissance est donc dynamique et adaptative.

Pour la fonction discrète, la décroissance est indépendante de la valeur de l'état précédent du système et dépend uniquement de la valeur de la différence ΔT . Cette fonction peut être utilisée lorsque le nombre d'itérations nécessaire pour atteindre un état d'équilibre peut être évalué avec une assez bonne précision.

III.2.3. Critère d'arrêt

Au fur et à mesure de la progression du processus d'optimisation par recuit simulé, on ne peut pas déterminer si la recherche a convergé vers la solution optimale, et pour y remédier on doit utiliser des indicateurs pour juger l'état de convergence de l'algorithme. Les indicateurs les plus répondue dans la littérature sont

Le taux des solutions rejetées lorsque cette grandeur devient trop importante, ce qui signifie que le nombre des configurations rejetées est élevé, alors on peut dire que l'algorithme n'évolue plus et il ne pourra éventuellement pas être capable de trouver des solutions meilleures. Pratiquement on peut fixer la valeur maximale de ce taux, ainsi juger la convergence par une simple comparaison.

La variance des coûts des solutions visitées plus ce dernier devient stable, alors l'algorithme expire. Donc la stabilité de cette grandeur peut indiquer la convergence du processus d'optimisation.

Température minimale simple, il suffit de fixer un seuil que régisse la plus basse température qu'on peut travailler avec.

III.2.4. Algorithme de recuit Simulé

En pratique, il n'est pas toujours possible de connaître a priori le nombre exact d'itérations nécessaire et le temps de calcul est limité, si bien que la convergence asymptotique ne peut être qu'approchée. A cause de cette approximation, l'algorithme de recuit simulé ne peut garantir d'atteindre le minimum global avec une probabilité égale à 1. Ainsi, pour augmenter la chance d'obtenir une solution la plus proche de l'optimum global, il est important de trouver les valeurs de compromis entre les paramètres de contrôle de l'algorithme.

En résumé, le recuit simulé utilise un double dynamique

1°) recherche de l'optimal à température fixée

2°) diminution par étape de la température

- **Initialisation** (soit X la solution courante, Xopt la solution optimale)
- Tant que (**critère d'arrêt non atteint**) faire
  - tant que le (**quasi équilibre non atteint**) faire
    - générer une solution voisine X'
    - si (**acceptation**(coût(X), coût(X'), Tcourante) alors
      - mise à jours de X et Xopt
  - fin tantque
  - **refroidir**(Tcourante)
- fin tant que.
- La meilleure solution est Xopt

Figure 22. Le recuit simulé est un algorithme progressant par palier de température à l'intérieur duquel un ensemble de configurations est généré et testé.

III.3. ALGORITHME LIEES AU RECUIT SIMULE (CAS CONTINU)

Dans cette partie nous allons parcourir quatre algorithmes reposant sur les principes du recuit simulé.

III.3.1. L'algorithme CSA (Classical Simulated Annealing)

Cet algorithme s'inspire directement du recuit simulé il ne se diffère qu'à deux niveaux

La manière dont on engendre une nouvelle configuration (un voisinage).en effet la configuration $Y = (y_1, y_2, ..., y_N)$ s'obtient de la configuration $X = (x_1, x_2, ..., x_N)$ on additionnant a la configuration X un vecteur aléatoire $Z = (z_1, z_2, ..., z_N)$ Dont chaque composant est tirée suivant une loi normale de moyenne nulle d'écart type variant avec la température.

Le schéma de température doit être à décroissance logarithmique

$$T_n = \frac{T_0}{\ln n}, n > 0$$

Equation 17. Schéma de température classique du recuit simulé.

III.3.2. L'algorithme FSA (Fast Simulated Annealing)

L'algorithme du FSA est le même que celui CSA sauf qu'il ce diffère de deux choses

La génération d'un Nouveau voisinage Le vecteur Z dans CSA ces composants suivaient une loi Normal, mais dans FSA les composants de Z suit une distribution de Cauchy de moyenne nulle et d'écart type en fonction de la température.

Quant au schéma de température il plus rapide, il correspond a une descente linéaire

$$T_n = \frac{T_0}{n}, n > 0$$

Equation 18. Schéma de température "rapide" du recuit simulé.

III.3.3. L'algorithme ASA (Adaptive Simulated Annealing)

Proposé par Lester Inber, cet algorithme ce différencie du recuit simulé et des autres méthodes du fait que la températures devient un vecteurs de même dimensions que l'espace de configurations, et que la variations de ces composants dépend de l'impact que rapporte les composants des configurations sur la fonction de coût, a cette ainsi que l'algorithme subit un schéma différent du recuit classique, qui prend trois étape, les voici

• Phase d'exploration

On va générer une nouvelle configuration, au quelle chaque composant suit une distribution (elle est identique pour tous les composants), qui est paramétrée par le composant du vecteur de température associer.

• Phase d'acceptation

On va tester chaque composant de la configuration Y_t , ou $t \in \Re^+$ désigne le temps de recuit. La nouvelle configuration et accepté avec la probabilité

$$p(x_{t+1} = y_t \mid x_0, ..., x_t) = \min\left[1, \frac{1}{1 + \exp\left(\frac{e(y_t) - e(x_t)}{T_t}\right)}\right]$$

X_t Est la configuration courante. Dans le cas ou la configuration Y_t n'est pas accepté, $X_{t+1} = X_t$ (X_{t+1} la prochaine configuration)

• Phase de reannealing (ou phase de rééchelonnement)

Ou chaque composant du vecteur de température va être modifié suivant l'impact de l'amélioration apportée par le composant lui associe dans la nouvelle solution, alors si cet impact est important, le composant va subir une faible variation, autrement il va subir une importante variation. Ce schéma intensifie la recherche dans des zones apportant de réelles améliorations, et diversifie la recherche dans le cas contraire.

• Phase de refroidissement

Le schéma de décroissance des températures suit une **loi exponentielle**.

PARTIE III. ACCELERATION ET PARALLELISATION

CHAPITRE IV. ACCELERATION SEQUENTIEL DU RECALAGE

IV.1. INTRODUCTION

Notre souci est d'améliorer le temps d'exécution du recalage par les méthodes de calcul séquentiels. Ce temps peut être prohibitif quand la taille des images s'accroît. Pour cela, il y a plusieurs méthodes d'accélération des calculs.

Nous allons suivre la stratégie d'Amdahl[1] pour accélérer le processus de recalage dans le sens « make the commun case faster » ou en français « rendre le cas le plus commun rapide ». Pour cela, on propose de voir un processus de recalage comme étant la succession de deux parties l'une est l'évaluation des solutions, l'autre englobe la totalité des autres traitements. Nos tests ont donné les résultats suivants

- L'évaluation des solutions est la partie la plus consommatrice de temps de calcul
- Le reste a été consommé par l'ensemble des autres traitements.

Ainsi donc, si on veut accélérer le recalage, il faut se concentrer sur la partie d'évaluation, et pour cela, il faut détailler ce qui se passe durant cette phase. En effet, l'évaluation d'une solution (qui n'est autre qu'une transformation) passe tout d'abord par :

- Le calcul de l'image transformée (cette étape consomme 96% du coût total du recalage),
- Ensuite le calcul de la similarité entre l'image consigne et l'image transformée (qui ne consomme que 03% du coût total).

Ces deux grandeurs nous démontrent que le coût n'est pas emporté par le parcours des pixels (<3%) mais par la complexité des fonctions qui sont nécessaires pour calculer l'image transformée (sinus, cosinus, multiplications,…etc.).

Puisqu'on ne peut pas se passer de ces fonctions, il ne nous reste que deux possibilités pour accélérer le recalage

- **Paralléliser l'étape d'évaluation.**
- **Diminuer la quantité d'informations à traiter.**

Dans cette partie on va s'intéresser surtout à la seconde possibilité, A cet effet, on a deux motifs pour ce choix, à savoir

[1] La loi d'Amdahl : une loi pour calculer l'accélération globale suite à une accélération relative (voir annexe 1).

- **Le premier cas** : c'est le fait qu'un volume important de l'image n'a rien à voir avec l'information qu'elle véhicule (fond de l'image par exemple) et, l'exécution sera améliorée par l'accélération physique « matériel » du recalage. C'est le partage des ressources de calcul selon des critères pour optimiser le temps et l'espace.
- **Le deuxième cas :** est le fait qu'on peut garder l'information de la ressemblance entre images rien qu'en traitant une partie des deux images et, l'exécution sera améliorée par l'accélération algorithmique du recalage. C'est le partage de l'espace de recherche selon des critères pour optimiser l'espace et le temps.

Imaginons cette possibilité, et qu'on peut en avoir une valeur rapprochée du coût d'une transformation dans un problème de recalage, rien qu'en traitant 10% des volumes des deux images, cela entraînera une accélération de 10 relative à l'étape d'évaluation, et une accélération globale de 9.17 selon Amdahl (voir Annexe I). Donc on peut diviser le temps total du recalage par 9.17 ce qui nous apporte un gain non négligeable.

IV.2. DIMINUER LA QUANTITE D'INFORMATIONS A TRAITER

Nous avons défini un opérateur d'extraction uniforme déterministe qui avait des résultats similaires aux résultats d'un zoom out des deux images, c'est pour cela qu'on va l'appeler zoom out.

Le zoom out de pourcentage P, d'une image dont la hauteur est H et la largeur est W, est une petite image dont le nombre de lignes est W'=W*racine (P/100), et le nombre de colonnes est H'=H*racine (p/100). En effet, les deux images doivent être proportionnelles, cette proposition est modélisée mathématiquement par l'équation :

W/H=W'/H'

Mais **W=nombre_de_pixels/H** et **W'=nombre_de_pixels'/H'**

Et **nombre_de_pixels'=nombre_de_pixels*P/100**

Par substitution on trouve les deux valeurs de **W'** et de **H'**.

Par la suite on définit deux variables PasH=H/H' et PasW=W/W' qui sont les pas de déplacement en hauteur et en largeur respectivement. L'image d'origine sera vue comme une matrice de W'*H' rectangles dont (StepH, StepW) sont la hauteur et la largeur de chaque rectangle.

Le pixel (i, j) de la petite image peut recevoir n'importe quelle combinaison des pixels du rectangle (i, j) de l'image d'origine. Nous avons choisi de prendre la valeur du niveau de gris du pixel du **coin supérieur gauche du rectangle**.

La réduction de l'image à recaler à P/100 de l'image origine, donc une résolution réduite ou une diminution d'échelle de la carte permet le des paramètres de calcul d'un taux de similarité maximum dans un temps réduit, que l'on appliquera sur la grande image et évaluer le temps améliorer sur la base des paramètres choisies.

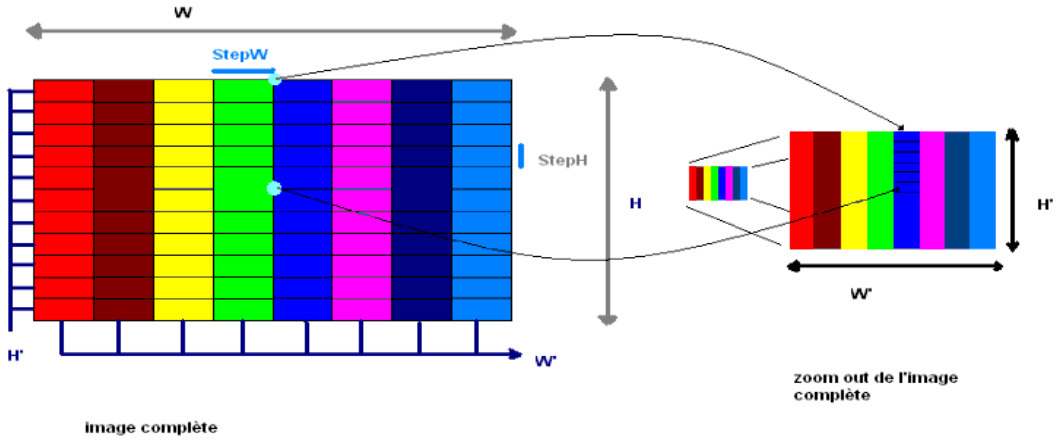


Figure 23. Schéma fonctionnel de l'opérateur Zoom out.

Voici des exemples des résultats de cet opérateur

Image complète	Zoom out de 30%	Zoom out de 10%
		
		Zoom out de 05%
		

Figure 24. Exemples d'images Zoomées.

IV.3. VALIDATION DE LA PROPOSITION

On doit démontrer tout d'abord, qu'on obtiendra des taux de similarité de même ordre de grandeur dans les deux cas de la similarité avec les deux images entières, et la similarité entre les deux ensembles de pixels prélevés à partir des deux images.

On va adopter le rapport de corrélation pour exprimer les similarités entre images, car c'est une valeur standardisée.

Pour valider cette approche d'accélération, il faut montrer que le prélèvement garde la même information ou une information similaire a celle d'origine. Et pour le faire, nous avons testé la similarité (au sens rapport de corrélation) de plus de 60 images zoomées 1%, 3%, 5% et 10% avec 5% et 10% on aboutira aux meilleurs écarts (moins de 0.01), avec 03% et 01% les écarts entre les similarités peuvent excéder 0.02 (voir ci-dessous).

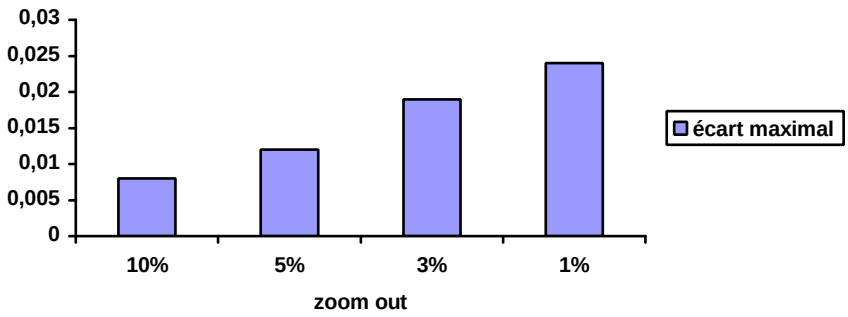


Figure 25. Ecarts maximals entre similarité d'images complètes et celle d'images zoomées.

Toutes les valeurs sont de même ordre de grandeur, ce qui a motivé la continuation dans cette voie.

IV.3.1. Schéma d'accélération

On propose de tester notre nouvelle approche de recalage suivante

- Calculer le zoom out des images consigne et celle à recaler.
- Effectuer le recalage sur les deux petites images.
- Effectuer un traitement sur la meilleure transformation délivrée, pour trouver la meilleure transformation du problème d'origine.

Les deux premières étapes sont claires, la nouveauté réside dans la dernière étape. On préfère ne pas détailler ce qui se passe durant cette phase pour l'instant, on le fera après les tests de la nouvelle méthode.

IV.3.2. Tests de validation (problèmes dont on connaît la solution)

Les tests ont été effectués en utilisant nos deux méthodes d'optimisation disponibles et déjà faites. Cependant, on ne citera pas explicitement leurs noms, pour ne pas donner des statistiques pour les deux, car tous ce qui nous importe est la qualité de la nouvelle méthode de recalage et pas autres choses.

Il faut dire que les tests ont été effectués sur un espace de recherche extrêmement large (800pixels*800pixels*6.28radian) en utilisant les configurations les plus lourdes, ceci dit que de telles configurations dans le cas de traitement complet prenaient quelques centaines de secondes.

Nous avons testé cette approche sur quelques dizaines de problèmes constitués de 6 images et leurs transformées, avec un zoom out de 1%, 3%, 5% et 10% avec 5% et 10% on aboutira aux meilleurs recalages inter petites images donc la grandeur 05% est largement suffisante pour le restant de nos tests.

Le recalage inter petites images dans tous les cas est parfait (rapport de corrélation >0.91). Lorsqu'on adopte la meilleure transformation dans le vrai problème où on traite les deux images complètes, on remarque dans tous les cas que l'angle de rotation est parfaitement aligné, mais les déplacements sont pratiquement loin de celles de la meilleure transformation. Cependant en les multipliant par les pas de déplacements (StepW et StepH) on obtient une solution très proche de l'optimale. Cela sera le traitement à faire pour adopter la solution rapide du recalage (voir ci-dessous).

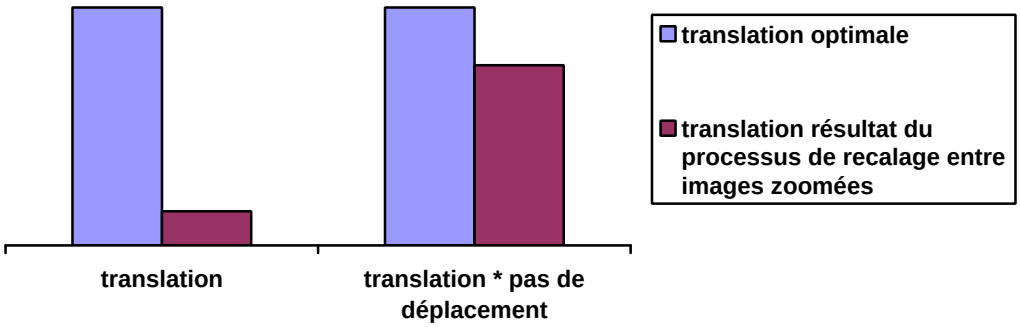


Figure 26. Graphe 1 adaptation des translations.

On n'est pas entrain de dire qu'on a résolu le problème de recalage original, car la marge d'erreur est de 20 pixels pour les deux translations. Mais avec cette approche on peut tirer un bon point de départ, donc de restreindre l'espace de recherche pour une autre méthode plus légère. Cette approche nous accorde un gain, en terme de temps de calcul, non négligeable remarquable (voir Graphe 2) (les durées de tests étaient inférieures à 2s).

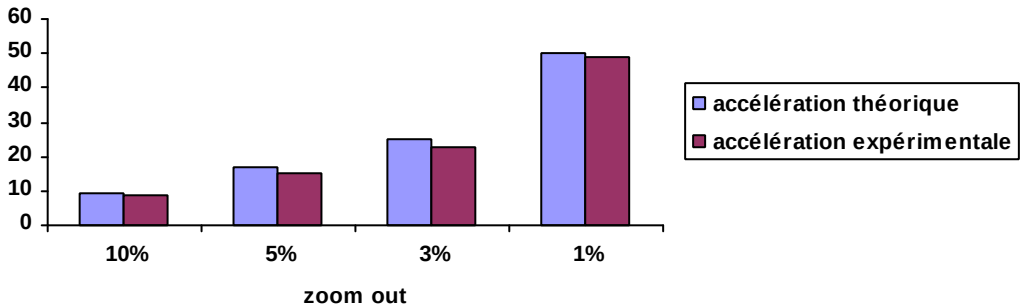


Figure 27. Graphe 2 : accélération théorique et celle apportée par un recalage entre images zoomées.

IV.4. UNE METHODE DE RECALAGE AUTOMATIQUE

Le processus de recalage suivant le schéma d'accélération par l'approche citée précédemment a convergé vers une solution mauvaise mais très proche de l'optimale. En effet, l'angle de rotation était parfaitement aligné, la marge d'écart entre les paramètres, de déplacement, ont été fixés à 20 pixels pratiquement.

Cela nous oriente vers une méthode de recalage intéressante, la voici

- **Effectuer le recalage (étape 2) sur les images qui ont reçu un zoom out de 5% (étape 1) pour bien cerner l'espace de variation de la meilleure transformation (étape 3)** durant cette étape on utilise un chercheur dont la configuration garantissant une convergence au niveau petites images, l'espace de variation des solutions de cet optimiseur doit être très étendu (nous avons choisi [-400, 400] pour les déplacements et [-3.14, 3.14] pour l'angle de rotation).
- **Cette étape est facultatif** effectuer le recalage sur les images qui ont reçu un zoom out de 50% pour une meilleure et une rapide restriction de l'espace de variation des paramètres.
- **Effectuer le recalage des images originales (étape 4)** en utilisant une configuration gagnante et légère sur un espace très réduit (étape 3).

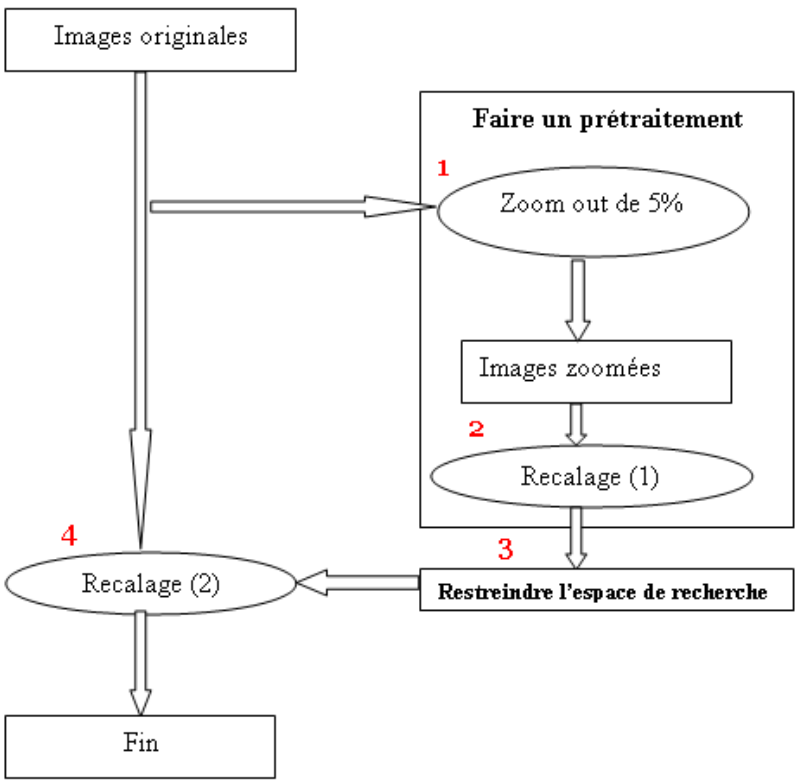


Figure 28. Schéma fonctionnel de la procédure de recalage automatique.

IV.5. RESULTATS APPORTES

Nous avons testé la nouvelle méthode de recalage avec le schéma d'accélération décrit auparavant sur les problèmes déjà traités par les méthodes classiques basées sur les algorithmes génétiques et le recuit simulé.

Les tests donnent les résultats suivants

- **Qualité** haute, de même ordre que celle garantie par les approches sans accélération.
- **Temps de réponse** une accélération considérable, l'accélération dépend étroitement du volume des images à traiter et du problème à résoudre. Le tableau suivant résume quelques résultats.

Résolution d'images	Coût calculatoire moyen (AGs)	Coût calculatoire moyen (RS)	Coût calculatoire moyen (AGs+ accélération)	Coût calculatoire moyen (RS +accélération)
128*128	50s	5s	07s	01s
256*256	70s	9s	10s	2s
700*800	100s	37s	16s	4s

Tableau 4. Comparaison entre les temps de réponse avant et après l'accélération.

IV.6. CONCLUSION

L'analyse des résultats de l'approche de recalage nous a orienté vers l'accélération de la partie d'évaluation des solutions dans le processus d'optimisation (qui est la partie la plus consommatrice). Deux choix étaient possibles pour accélérer cette partie : la parallélisation de calcul, ou la diminution de la quantité des données à traiter.

Dans cette partie nous avons essayé d'accélérer le processus de recalage en diminuant la quantité des données à traiter par un opérateur de zoom out, nous avons remarqué qu'on peut garder l'information utile rien qu'en traitant 5% des deux images, cela entraînera une accélération de 20 relative à l'évaluation des solutions, et une accélération globale de 16.8 (selon Amdahl) qui nous accorde un gain non négligeable. Mais durant les tests et les traitements supplémentaires qu'on a fait, nous avons remarqué qu'en réalité cette approche ne peut optimiser qu'un seul paramètre (l'angle de rotation) tandis que les déplacements avaient toujours de l'écart par rapport à la meilleure solution.

Cela signifié que l'algorithme a convergé vers une mauvaise solution mais très proche de l'optimale, cette solution sera plus tard comme point de départ pour d'autres méthodes d'optimisation plus légère. Cette approche va définir une méthode de recalage automatique rapide et efficace.

En fin, on peut valider les deux modèles avec accélération, tout en acceptant le schéma du recuit simulé accéléré qui donne des bons résultats.

CHAPITRE V. PARALLELISATION DU RECALAGE

V.1. INTRODUCTION

Dans le chapitre précédent nous avons vu l'accélération séquentielle des calculs relatifs au processus de recalage à savoir les méthodes d'optimisation que sont le recuit simulé et les algorithmes génétiques. Nous avons vu que nous pouvions réduire considérablement les temps de calcul. Néanmoins, toute autre méthode qui permettra de réduire davantage les coûts de calcul sera étudiée. Dans ce contexte, les architectures parallèles peuvent être d'un grand secours pour la parallélisation d'un certain nombre de calculs.

Nous allons dans ce chapitre faire un tour d'horizon sur les architectures parallèles et voir comment on peut exécuter sur une telle architecture les méthodes d'optimisation qui ont fait l'objet de nos tests. L'accent sera mis sur la partie des calculs qui peuvent et qui doivent être réparties pour obtenir un gain substantiel.

V.2. PARALLELISATION DES ALGORITHMES GENETIQUES

La version séquentielle des AGs a été utilisée avec excellence dans plusieurs applications de divers domaines, mais il existe des problèmes de complexité supérieure et que même les puissantes procédures séquentielles ne peuvent donner des résultats satisfaisants. Donc on cherche toujours à réduire le coût en termes de temps de calcul ou temps de réponse. La parallélisation de traitement est une solution par excellence pour remédier à ce problème. Le processus d'évolution génétique possède une nature parallèle, l'évaluation et la mutation des individus se fait d'une manière indépendante, les seuls traitements nécessitant une quelque sorte de synchronisation sont la sélection et le croisement.

Les algorithmes génétiques parallèles (AGsP) est donc la solution.

Ce genre d'algorithmes ne représente pas seulement une extension des AGs classiques, mais ils représentent une nouvelle gamme pourquoi pas une nouvelle classe d'algorithmes dont l'exploration est un peux plus particulière, plus différente.

Une bonne parallélisation des algorithmes génétiques doit préserver toutes les caractéristiques des AGs séquentiels, aussi elle ne doit jamais introduire un nombre important des paramètres supplémentaires qui influent sur la qualité de convergence de l'algorithme, finalement elle doit minimiser (éliminer si c'est possible) tout statut de synchronisation, et de communication.

Plusieurs travaux ont été lancés et avaient pour objectif de fournir une classification satisfaisante des AGsP, mais apparemment personne n'a réalisé clairement et pleinement ce travail. Cette classification doit répondre aux questions suivantes

Comment la fitness est évaluée ? Comment la mutation est appliquée ?
Si une ou plusieurs (sous) populations sont utilisées ?
Dans le cas de populations multiple, quelle est la politique de migration utilisée ?

La sélection est elle locale ou globale ? Les réponses à ces quatre questions élaborent huit classes pour les AGsP qui peuvent être réduit à six, citons les plus intéressantes ;

V.2.1. Parallélisation maître/esclave (GPGA, global master/slave parallel genetic algorithms) :

L'évaluation distribuée des fitness des individus peut réduire le coût de la recherche. En effet, l'idée vient du fait que le calcule des fitness et les mutations peuvent se faire indépendamment. Dans ce cas là on dispose d'une architecture maître/esclave, le maître assume le traitement des fitness des individus aussi la sélection des individus. Quant à eux, les esclaves évaluent les fitness et effectuent les mutations, éventuellement assumant le croisement.

Le maître commence par le partage de la population en plusieurs sous populations (au nombre des processeurs esclaves). En suite les esclaves évaluent la fitness de chaque individu, d'appliquer, éventuellement, les opérateurs de reproduction, en suite envoyer les informations requises au maître pour effectuer la sélection et ainsi de suite. Dans une nouvelle version propose que le maître ne fait que générer une population initiale, l'esclave fait évoluer la population en utilisant une sélection par tournoi, cette approche bien qu'elle ne nécessite aucune communication ni transmission, elle n'a de sens en absence d'une mémoire partagée.

La communication maître/esclave peut être synchrone et asynchrone. Dans le premier schéma le maître doit attendre que tous les esclaves accomplir leurs taches, ce schéma préserve la philosophie et les caractéristiques des AG séquentiels, mais avec un peut d'accélération de traitement. Dans un cas de communication asynchrone le processeur maître attend une certaine durée (donnée) en espérant que les esclaves ont accomplis leurs taches, en suite il effectue son traitement.

Ce genre de parallélisation n'a d'utilité que si le coût d'évaluation des fitness dominera la complexité du problème, autrement et suivant la loi d'Amdhal[2], elle n'apporte aucune (ou faible) accélération globale.

V.2.2. Parallélisation par sous populations statiques avec opérateur de migration

Cette parallélisation consiste à diviser la population en plusieurs sous populations ou « îlots » sur des processeurs, ces îlots sont géographiquement isolés (séparés) et un individu n'évolue que dans son propre sous population, à moins qu'il puisse recevoir un nouvel opérateur de « migration ». Cet opérateur permet qu'un individu migre d'un îlot vers un autre. Plusieurs stratégies de migration sont possibles, par exemple la migration peut se faire entre n'importe quels îlots, ou bien elle ne peut se faire qu'entre deux îlots voisins. Mais il est évident que cet opérateur entraîne un surcoût, et il faut bien choisir la stratégie adéquate pour minimiser ce coût, qui est contrôlé par plusieurs paramètres citons par exemple le taux et la fréquence de migration, la topologie de la liaison inter îlots.

L'absence de migration entraîne l'évolution indépendante des îlots, dans ce cas chaque sous population converge vers une solution optimale locale. Ce qui nous fait gagner le coût de communications, Cependant, l'évolution indépendante des sous populations provoque une mauvaise exploration de l'espace de recherche. Dans le sens où il n'y aura pas de diversité intra population ce qui conduit sans doute à une convergence inter population prématurée.

[2] Loi permettant de mesurer l'accélération globale. (Voir annexe 1)

Cet opérateur est très important vue ça capacité d'engendrer une diversité dans les îlots ce qui conduit à une meilleurs exploration de l'espace de recherche.

La migration peut être synchrone ou asynchrone, la première à l'avantage de simplicité et de facilité de mise en œuvre, cependant le coût de communication s'accroît. Tandis que la deuxième nécessite une unité supplémentaire pour le traitement de transfert des informations, mais elle nous fait gagner du temps (relatif à la synchronisation).

En fin, ce genre de parallélisation peut engendrer deux autres schémas suivant le nombre des îlots relativement la taille de ces derniers en deux autre schémas

Les AGsP à grain gros (les AG distribués):

Ce schéma est caractérisé par un nombre réduit d'îlots, donc une taille importante de chaque îlot, aussi caractérisé par un faible taux de migration, ce schéma favorise une implémentation sur une machine MIMD[3] ou sur un réseau de stations d'où le nom AG distribués.

Les AGsP à grain fin

Contrairement aux algorithmes génétiques parallèles à grain gros, les algorithmes à grain fin se caractérisent par un nombre important d'îlots de petite taille qui peuvent arriver à un seul individu par îlot [SAL01]. Pour éviter une convergence prématurée de recherche, le taux de migration doit être important.

Plusieurs paramètres supplémentaires sont à régler avec prudence dans ce schéma pour garantir le bon fonctionnement de l'algorithme, les plus cruciales sont les instants de migration et la fréquence de migration. Intelligemment les migrations doivent être retardées un certain moment au début pour laisser évoluer les îlots (la diversité n'apporte pas grand-chose à ce stade). Au fur et mesure la fréquence de migration doit s'accroître pour créer plus de diversité car le processus d'optimisation génétique évoluent très rapidement. Les instants de migration sont contrôlés par l'observation d'une autre grandeur. Cette approche semble être intéressante.

Ce schéma peut être implémenté sur des architectures parallèles, qu'elles soient fortement couplées (avec mémoire commune), ou faiblement couplées (sans mémoire commune). Mais il a l'allure d'être conçu pour être de la « taille » d'une architecture faiblement couplée, un réseau d'ordinateurs par exemple.

[3] MIMD: multiple instructions for multiple data.

V.2.3. Les algorithmes génétiques hiérarchiques

Un schéma maître/esclave à plusieurs niveaux, les esclaves évoluent des sous populations et une communication esclave/maître est organisée périodiquement pour l'échange des individus. Le maître peut être un esclave d'un maître de niveau supérieur et applique ainsi la même stratégie et ainsi de suite. L'échange des individus peut créer la diversité nécessaire pour garantir une bonne exploration du paysage de recherche.

V.2.4. Parallélisation par sous population sans opérateur de migration

L'idée est la même que l'avant précédente (V.2.2), cependant on ne dispose pas d'un opérateur de migration. Dans ce schéma la population est divisée en plusieurs îlots, mais contrairement au schéma précédent (V.2.2), ces îlots ne sont pas isolés. Des individus peuvent appartenir à plusieurs îlots, et peuvent être croisés ou mutés plusieurs fois en une génération.

Ce schéma représente un excellent choix, si on dispose d'une machine parallèle à mémoire globale partagée.

V.2.5. Algorithme génétique massivement parallèle

Un cas particulier d'une parallélisation grain fin, taillé sur les mesures d'une machine massivement parallèle. A raison d'un individu par îlot (processeur), la migration ne peut se faire n'importe comment mais suivant la connexion inter processeurs de la machine, donc un individu ne peut se connecter qu'avec son voisin. Il est clair que le taux de communication est important ce qui prohibé l'implémentation d'un processus d'optimisation sérieux sur une architecture dont le coût de communication est important (un cluster d'ordinateurs par exemple).

V.2.6. Ilots ou sous populations dynamiques

Ce schéma peut être considéré comme modification du schéma de parallélisation gros grains, ou encore l'implémentation de ce dernier sur une architecture maître/esclave pour compenser l'absence de la migration. Donc pas de migration dans ce schéma, le partage des informations se fait par réorganisation dynamique des îlots, et c'est la charge du maître. Quant aux esclaves, ils évoluent une population qui se change dynamiquement. Avec une mémoire globale partagés et un coût d'accès aux individus négligeable (machine MIMD), ce schéma peut donner des résultats excellents. Dans le cas contraire et surtout si la taille des chromosomes (à transmettre) est trop grande, encore si le coût de communication soit trop élevé (un cluster d'ordinateurs) alors cette parallélisation est prohibitive.

V.3. LES ARCHITECTURES DES ALGORITHMES GENETIQUES PARALLELES

Dans certains cas, comme il l'est pour le recalage d'images, le déroulement de l'algorithme génétique prend du temps avant de pouvoir afficher un résultat. Cela se traduit par un temps de calcul trop lourd dû dans notre cas aux opérations répétés, sur une très grande population d'individus. Pour pallier à ce problème une stratégie de parallélisation de l'algorithme s'impose, d'autant plus que l'algorithme génétique est particulièrement adapté à ce procédé en raison de sa structure, avant d'aborder les algorithmes génétiques parallèles proprement dit, nous ferons d'abord une introduction dans ce domaine de l'informatique en définissant les outils et les types d'architectures nécessaires.

V.3.1.1. Types d'architectures

Le parallélisme évoque la mise en œuvre d'une architecture judicieuse capable de traiter des opérations élémentaires en simultané en se basant sur un montage spécifique des unités de calcul. Ainsi un même problème pourra être résolu selon différentes manières qui dépendent bien entendu de la parallélisation de l'algorithme en elle-même. On distingue trois grandes approches:

- Prendre les avantages de la structure de l'algorithme séquentiel et l'adapter pour qu'il soit déroulé de manière parallèle.
- Faire la transformation d'une solution parallèle déjà existante d'un problème puis l'ajuster pour la résolution actuelle.
- Développer une architecture parallèle nouvelle qui résout le problème.

Pour les algorithmes génétiques par exemple, on utilise généralement la première approche, bien qu'il existe des architectures se basant sur la troisième, intégrant des programmes parallèles pour la résolution des problèmes (on peut citer par exemple la machine Paragon d'Intel).

On effectue aussi une classification des ordinateurs selon la multiplicité des flux d'instructions et des données disponibles. On distingue trois types de configurations fondamentales:

SISD; Single Instruction stream Single Data stream. La machine traite un seul flux d'instructions et de données. C'est l'ordinateur classique de Von Neumann, en plus d'effectuer des traitements séquentiels on peut les monter en cascade (pipeline) pour les applications utilisant cette configuration. Ex : PC, station de travail.

SIMD; Single Instruction stream Multiple Data stream. L'ordinateur traite un seul flux d'instructions et plusieurs flux de données. Elle est constituée d'une unité de contrôle UC qui supervise plusieurs unités de traitement UT qui exécutent les instructions simultanément. L'UC commande aussi le réseau synchrone pour l'accès des UT à la mémoire. Ex : Maspar.

MIMD; Multiple Instruction stream Multiple Data stream. Exécution de plusieurs flux d'instructions et plusieurs flux de données. Chaque UT possède sa propre unité de contrôle, ainsi les processeurs peuvent fonctionner d'une manière totalement asynchrone offrant ainsi une possibilité d'exécution de programmes différents. Une structure MIMD est dite intrinsèque si elle implique des interactions entre différentes unités de traitement. Elle est dite fortement couplée si les interactions entre les UT sont assez importantes. Dans le commerce, la majorité des architectures sont faiblement couplées. Ex : Paragon.

Le mode de programmation SPMD issu de la structure SIMD offre la possibilité d'exécuter un même programme diffusé par l'unité de contrôle sur toutes les unités de traitement. Les données peuvent provenir de flux distincts. On distingue ainsi des architectures SIMD/SPMD et MIMD/SPMD.

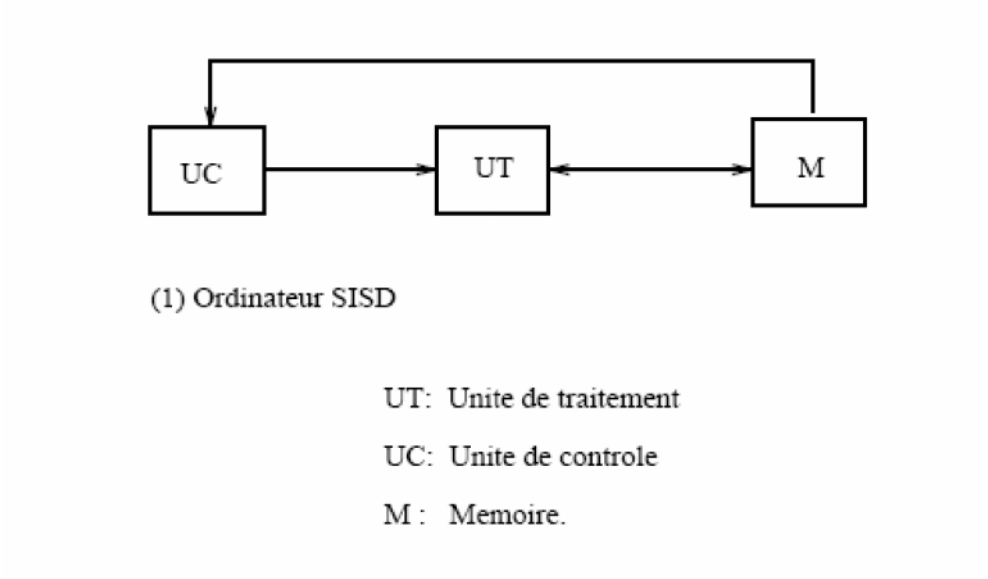


Figure 29.*Schéma d'un calculateur SISD en cascade*

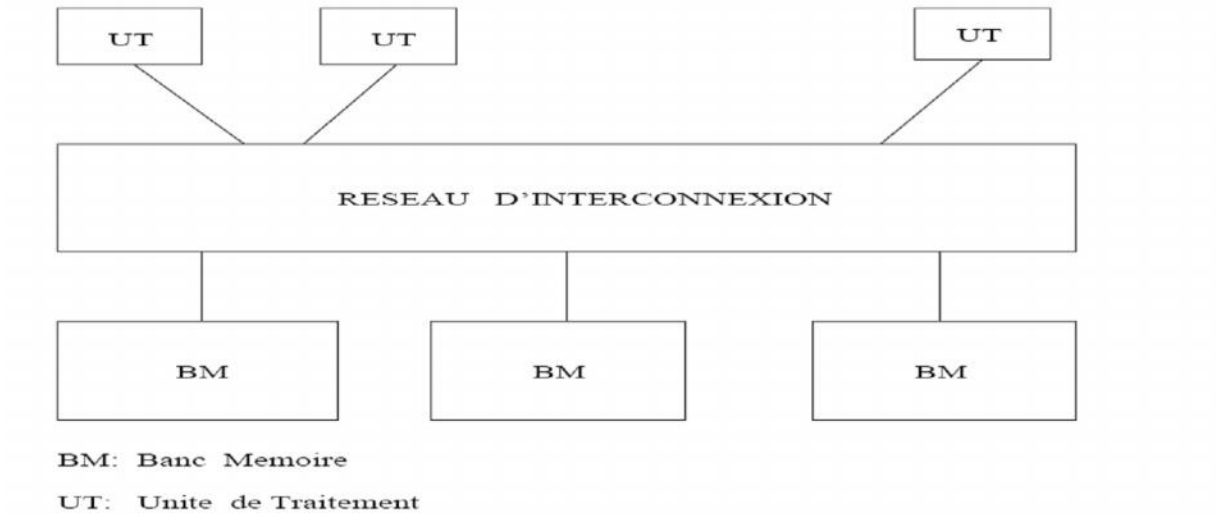


Figure 30.*Schéma d'un calculateur SIMD*

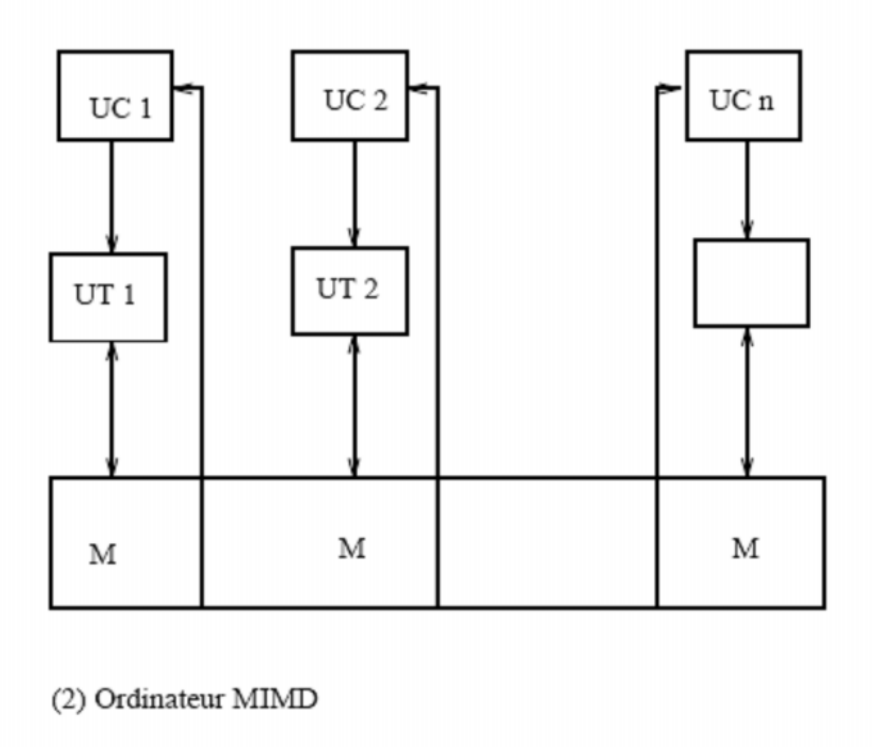


Figure 31.*Schéma d'un calculateur MIMD*

Pour permettre aux UT d'opérer l'architecture doit pouvoir laisser libre accès aux données. Ce problème est de nature combinatoire pour les SIMD. En effet lors de la lecture ou de l'écriture d'un mot sur la mémoire par un processeur, il ne faudrait pas qu'il y ait des conflits d'adresses. Un moyen d'y remédier est de partitionner la mémoire centrale en autant de champs que de processeurs. Si M est le nombre de champs alloués dans la mémoire et T le nombre de processeurs, alors la relation $M \geq T$ doit donc être vérifiée avant l'exécution de l'algorithme.

On parlera de parallélisme de contrôle lorsque celui-ci porte sur les tâches qui peuvent s'exécuter indépendamment les unes des autres. Lorsque l'on trouve des données qui peuvent être traitées simultanément, le parallélisme est dit de données. Un même algorithme séquentiel peut ainsi conduire à plusieurs versions parallèles, relativement à la synchronisation des processeurs, au mode d'accès des données, au découpage du problème en tâches et à l'affectation de celles-ci aux différentes unités de traitement. Certaines versions seront alors mieux adaptées à une structure d'ordinateur donnée. Une machine SIMD, par exemple, a du mal à traiter un algorithme asynchrone, alors que les problèmes de synchronisation des processeurs pour un ordinateur MIMD sont purement matériels.

V.4. L'ALGORITHME GENETIQUE PARALLELE :

La parallélisation d'un AG a pour but principal d'augmenter son efficacité en réduisant le temps de calcul. Les différents calculs sont distribués sur un ensemble de processeurs pouvant communiquer via un réseau d'interconnexion (hypercube, réseau maillé, etc.) Actuellement, les différentes architectures permettent de classifier l'ensemble des algorithmes génétiques distribués en trois types :

(a) Modèle parallèle standard ou modèle centralisé

L'évaluation, le croisement et la mutation se font en parallèle, tandis que la sélection est séquentielle donc centralisée. Ce modèle n'est pas flexible puisque le coût de communication croît exponentiellement en fonction de l'effectif de la population.

(b) Modèle de migration

La population est divisée en sous populations de même taille. Chaque processeur exécute l'algorithme standard sur la sous population qui lui est affectée. Pour propager les meilleurs individus entre les sous population, les nœuds échangent périodiquement des données. Les meilleurs individus reçus vont ainsi remplacer les plus mauvais de la population locale. Cet échange empêche chaque nœud de tomber dans un optimum local. La fréquence des échanges et le nombre des individus affectés sont des paramètres de l'algorithme. Dans cette approche, le parallélisme inhérent aux algorithmes génétiques (partage en sous domaines) n'est pas totalement exploité, le traitement d'une sous population étant séquentielle. Ce modèle est cependant intéressant dès lors que le nombre de processeurs disponibles est plus petit que la taille de la population désirée

(c) Modèle de diffusion

Chaque processeur traite un seul individu et toutes les opérations se font en parallèle. East et Macfarlane ont montré que cette approche donne en général les meilleures solutions avec un minimum de temps dans différents schémas d'exécution.

V.4.1. Conclusion sur le parallélisme des AGs:

Durant ce paragraphe on a essayé d'énumérer les principales classes (schémas) des algorithmes génétiques parallèles, qui sont bien nombreux et de diverses natures et de variété d'idées et de principes. Ils ne sont pas les seuls bien entendu mais on les a jugés les plus importants, de plus le reste semble être inclus dans d'autres classes.

Notre travail doit être conçu et réalisé pour tourner sur un réseau de stations de travail (éventuellement hétérogènes), ce dernier peut être vu comme étant une architecture parallèle faiblement couplée (sans mémoire partagée) dont la communication inter unités de traitement est de coût non négligeable. De première vue le parallélisme à gros grain semble le seul à être adopté considérant le faible taux de communication inter îlots dans ce schéma (relativement aux autres schémas).

V.5. ACCELERATION DE L'ALGORITHME DU RECUIT SIMULE

Différentes possibilités sont offertes pour accélérer la recherche, d'une part on jouant sur les paramètres de l'algorithme (le schéma de la température…), d'une autre part on va faire paralléliser l'algorithme lui même.

V.6. LE RECUIT SIMULE EVOLUTIONNAIRE PARALLELE (ESA) :

Comme on l'a montré au chapitre précédent, l' « ESA » permet, plus efficacement, d'éviter les piéges des minimas locaux. Donc, lorsque l'algorithme manipule une certaine solution assez longtemps, il garanti un résultat plus satisfaisant, il a aussi été signalé qu'une population de plus grande taille améliore la performance de la recherche. Cela dit, il parait évident que ces deux dernières propriétés ne soient pas compatibles. C'est-à-dire :

Supposons que nous ayons une population P et un nombre donné d'évolutions N où d'évaluations, et soit « ci » le nombre sélectionné pour la $I^{ième}$ solution. L'idée est d'avoir : **ci ≈ |P|/N** pour donner à l'algorithme la possibilité de manipuler suffisamment chaque individu. D'un autre côté, si « ci » nécessite « ti » par rapport au temps de calcul globale du processus (CPU time), le nombre totale de sélections dans la population « cp » et surtout le temps de calcul globale « tp » s'exprimeront :

$$c_p = \sum_{i=1}^{|P|} c_i \quad \text{et} \quad t_p = \sum_{i=1}^{|P|} t_i$$

L'opposition entre la taille de la population, qui apporte la diversité, et le temps de calcul peut être résolue par l'utilisation d'une population réduite. Contrairement à l'algorithme génétique, on devra donc travailler sur une population réduite pour que l'avantage de la diversité soit joint au recuit simulé amélioré sans perdre en rapport efficacité/temps de calcul. La réduction de la taille de population détériore généralement le spectre de la diversité. C'est pour palier à ce nouvel obstacle que l'on fait appel à présent à la structure de la programmation parallèle ou distribuée. L'idée est de distribuer le système à travers la distribution d'une population large sur l'ensemble des ressources machines. Pour réussir cela, on doit créer des agents d'identité que l'on exécutera sur les différentes ressources pour chaque sous partie de la population.

Comme on le sait, il existe deux façons d'implémenter un système en exécution parallèle

- La première, en utilisant la parallélisation physique, partitionne l'ensemble des données et les distribue sur des exécutions parallèles sur un ensemble de machines ou de processeurs. La plupart des systèmes évolutionnaires parallèles utilisent cette méthode.

- La seconde, où la parallélisation s'applique à l'algorithme lui-même, est un peut plus compliquée. C'est ce que l'on appelle une parallélisation algorithmique, qui est développé en générale pour des architectures parallèles du type SIMD et MIMD..Etant donnée que la parallélisation algorithmique d'un recuit simulé serait pour le moins très difficile à réaliser, on choisira la parallélisation du système au sens physique.

V.6.1. Le Recuit Simulé Evolutionnaire Parallèle sur un système multi agents:

Pour l'implémentation du « ESA », une architecture de machines parallèles est requise. Les réseaux spécifiques tels que les machines massivement parallèles, n'étant pas accessible en générale, on considérera l'environnement le plus commun et le plus facile d'accès, celui des WAN/LAN (Wide/Local Area Network of computers). La ressource distribuée de machines (DRM) est une infrastructure qui fourni un environnement de résolution de problèmes parallèles basée sur des agents mobiles. C'est l'infrastructure parallèle du logiciel DREAM² (Jelasity, Preuβ et Peachter, 2002), qui à été développée pour une résolution à travers un algorithme évolutionnaire distribué sur un réseau massif de nœuds sur Internet. Le but principal de ce système est la résolution de problème basée sur des systèmes de multi agents, qui exécutent des algorithmes évolutionnaires. Le système est un réseau de ressources qui fonctionne comme un réseau d'échange en nœuds (peer-to-peer) distribué sur une architecture parallèle de machines. Chaque nœud ne possède qu'une connaissance partielle du problème et fonctionne comme le maître de l'ensemble des agents exécutés sur une machine. L'environnement possède donc d'excellentes fonctionnalités pour le développement de ce type d'application compte tenu de la très bonne communication entre les agents et sa mobilité limitée.

La distribution du processus évolutionnaire sur les ressources d'après DREAM se fait selon de model des îlots (island model). Les îlots sont construits et dotés de plusieurs propriétés, données et algorithmes, puis sont distribués sur le réseau DRM. L'environnement DRM est développé sur la base d'une programmation à traitements multiples en Java (C++). Les îlots y sont lancés en MIR et sont dotés d'un système de transmissions (Message Passing System, MPS) qui utilise une connexion selon les protocoles TCP/IP. Etant donné que l' « ESA » est d'une nature évolutionnaire, et donc implicitement parallèlisable, le DRM permettra facilement de développer l' « ESA » selon un model d'applications sur îlots. Chaque îlot est doté d'un algorithme « ESA » identique et d'une sous partie de population. Il reste à définir le critère de sélection, la règle de remplacement et la stratégie de transfert des résultats atteints par les îlots. Les îlots feront évoluer leur sous population avec leur « ESA » et auront la possibilité de faire migrer une solution vers d'autres îlots selon une certaine règle.

Au sens de la parallélisation, il est préférable d'avoir un temps de calcul global, (tp) réduit pour le même nombre de sélections (cp), ceci pour une seule sélection. Le temps de calcul global pour l'ensemble du processus sera donc réduit linéairement proportionnellement au nombre d'agents a. Selon les notations utilisées auparavant, P sera partitionné en a parties, P1, P2,…,Pa. En assignant chaque partition à un îlot, le nombre de sélection étant le même partout, on obtient :

$$c_p = \sum_{i=1}^{a} c_j \ , \ c_j = \sum_{i=1}^{|P_j|} c_i \ \text{et} \ \ t_j = \sum_{i=1}^{|P_j|} t_i$$

Où cj et tj sont le nombre total de sélection et le temps de calcul total dans l'îlot j. Alors, cp ≈ a × cj et = Max {t1, t2,…, ta}.

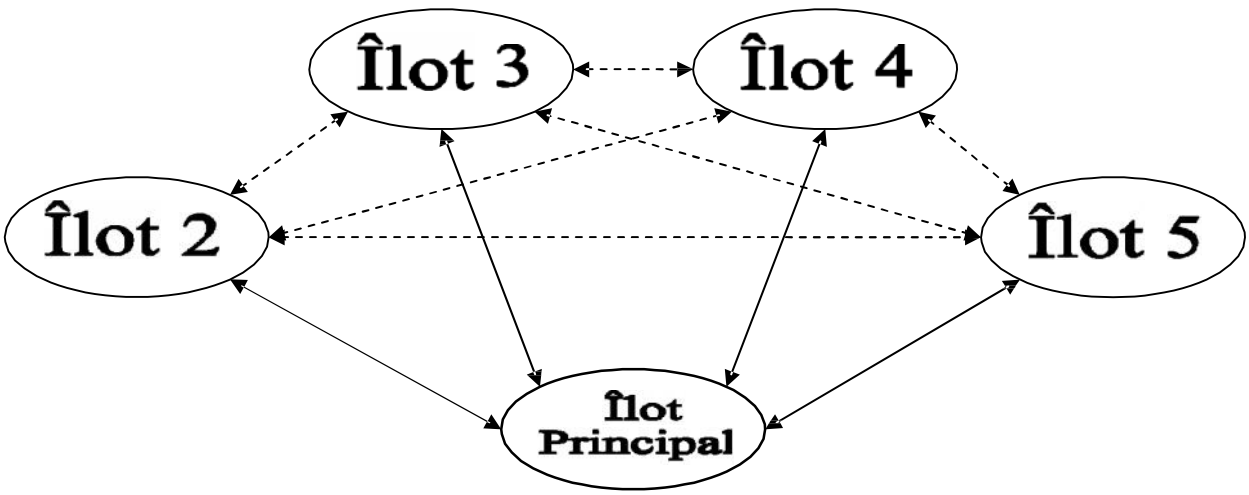
Donc, pour le même cp, on obtient un tp plus court, même en considérant le temps de communication entre les îlots. On peut clairement conclure que l'exécution parallèle des agents réduit le temps nécessaire pour atteindre l'optimum global.

V.6.2. Conception du système multi îlots :

Pour un problème particulier, P est traité par un nombre N d'itérations. Chaque individu sera plus ou moins manipulé cj fois. Pour préserver les équités de traitement entre l'ensemble des individus, on distribuera P sur les « a » îlots équitablement, et on déterminera le nombre d'itérations à appliquer. Chaque îlot traitera donc une sous population de taille |P|/a en N/a itérations. Un des îlots sera désigné comme l'îlot principal et aura la charge de collecter et de fournir les données et les 'meilleurs résultats' à certaines périodes. Comme le montre le schéma 1. Les îlots sont tous connectés entre eux. Les flèches discontinues représentent les liaisons inter îlots, et les continues représentent les liaisons entres l'îlots principal et les autres. L'îlot principal est toujours celui duquel est lancé le système. Le système étant très malléable, il peut être élargi en y connectant d'autres îlots très facilement.

V.6.3. Spécification des îlots :

Chaque îlot employé dans le système doit être identique aux autres, à l'exception de l'îlot principal qui nécessite plus de puissance pour administrer le système en parallèle. Une solution est candidate à la migration vers un îlot aléatoire après un nombre prédéfini d'itérations en utilisant les systèmes d'échanges et de prise en charge fournis par le DRM. La solution candidate à la migration est sélectionnée aléatoirement dans la population. Lorsqu'un îlot reçoit un immigrant, il l'accepte selon la règle de remplacement en décidant de l'échanger avec un de ses individus ou non. Après la période prédéfinie, chaque îlot envoi sa meilleure solution à l'îlot principal, qui les compare entre elles et détermine la meilleur de toutes. A son lancement, le DRM assigne l'îlot principal, puis défini l'ensemble des autres îlots présent dans le système, leur fournissant l'adresse de l'îlot principale. A chaque îlot est ensuite affecté une sous partie de la population, toutes totalement indépendantes entre elles, et selon un possible critère de diversité.



V.7. ACCELERATION PAR CHOIX DE PARAMETRE

a. Le schéma de décroissance exponentielle qui est de la forme

$$T_n = T_0.\alpha^n, avec\_n \in \mathrm{N}^+ \text{ Et } 0<\alpha<1$$

b. Modification de Voisinage

- À Haute température presque toutes les configurations sont admissibles, on définit donc un voisinage très élargi qui touche pratiquement tous l'espace de recherche, pour permettre à l'algorithme d'évaluer plus dans divers point de l'espace.
- À basse température le voisinage sera restreint aux configurations ne variant que très peut par rapport à la configuration courante.

V.8. CONCLUSION

Un problème d'optimisation combinatoire, est un problème dont on ne peut ni énumérer son espace de recherche ni de trouver une solution optimale globale en un temps raisonnable, à cette effet qu'on a développé des méthodes brutes pour résoudre ces problèmes difficiles, se sont des méthodes de recherche aléatoire qui cherchent de manière aléatoire ou guider une solution approchée de l'optimum globale en un temps réduit.

Dans ce chapitre on a essayé de couvrir tous les éléments qui constituent un algorithme génétique. Une preuve de convergence a été élaborée, malheureusement cette preuve ne concerne que la version canonique des algorithmes génétiques, mais cela n'a pas empêché l'attestation typique et expérimentale de leur excellence (en termes de qualité de convergence). En effet ils ont arrivé à combiner une meilleure exploration de l'espace de recherche et une excellente exploitation des informations acquises durant le processus d'exploration.

Comme toute approche paramétrique, les paramètres d'un algorithme génétiques doivent être bien choisis, pour arriver à l'excellence désirée. On Commence par le codage des solutions qui est trivial dans un cas de manipulation des grandeurs discrète, dans le cas contraire (manipulation des grandeurs réelles qui est notre cadre d'étude) on peut procéder à une discrétisation (précision limité) ou encore de remonter le niveau et de travailler directement sur le niveau phénotypique. En suite de bien définir les opérateurs de sélection et de la reproduction et de leurs taux, arrivant aux finalisations.

Comme nous l'avons remarqué aussi que les nouvelles versions des algorithmes génétiques peuvent travailler sur le niveau phénotypique tout comme les stratégies d'évolution ou l'évolution différentielle donc on peut utiliser leurs opérateurs pour enrichir le répertoire des AGs.

Pour le Recuit Simulé est bien encadre par un fondement mathématique qui prouve sa convergence vers l'optimum global. Le recuit simulé est une méthode approchée (cherche une solution approchée) globale (qui n'est pas sensible aux optimums locaux) inspirer de la thermodynamique, et c'est l'une des méthodes la plus populaire dans le comité scientifique.

En ce qui concerne la parallélisation, bien que le choix semble être évident, on croit qu'il ne faut pas décider avant d'analyser correctement le coût de chaque opérateur, c'est à ce stade là qu'on peut choisir une méthode de parallélisation adéquate (suivant la loi d'Amdhal (voir annexe 1)).

PARTIE IV CONCEPTION ET MISE EN ŒUVRE

CHAPITRE VI. CONCEPTION

VI.1. INTRODUCTION

A travers la partie précédente, nous avons constaté que le recalage d'images était tributaire d'un nombre important de tâches ; ces dernières sont d'une très grande utilité pour le recalage des images satellitaires d'une façon générale et à la télédétection d'une façon particulière. Le recalage devient donc, un problème central et primordial pour le traitement d'images satellitaires.

Le recalage étant la maximisation de similarité entre images, devient un problème d'optimisation combinatoire, pour lequel on peut utiliser n'importe quelle méthode d'optimisation pour le résoudre. Dans ce cas il est important de définir la méthode à adopter ?

Par ailleurs, il y'a lieu de rappeler que les problèmes d'optimisation combinatoire sont posés depuis longtemps. La littérature fournit plusieurs méthodes pour les résoudre, parmi lesquelles se trouvent les métaheuristiques. Nous avons étudié succinctement les notions fondamentales de deux familles de métaheuristiques (les algorithmes évolutives et par voisinage), et en détails deux méthodes évolutives (algorithmes génétiques et algorithmes génétiques quantiques) et une méthode par voisinage (recuit simulé). Chaque méthode se caractérise par sa manière propre de compromis exploration/exploitation, ce qui défini le critère d'évaluation la puissance d'une méthode d'optimisation.

Voila deux domaines ou deux maillons totalement distincts, dont la liaison réalisera la tâche requise, à savoir, le recalage sur la base de la maximisation de similarité (par métaheuristiques).

Dans ce chapitre on va faire lier ces deux domaines, pour construire une plateforme complète qui a pour but de réaliser le recalage ainsi que l'analyse correcte des résultats, c'est une simulation informatisée.

VI.2. CONCEPTION

Le modèle formel présenté dans le chapitre de l'état de l'art, définit le recalage comme étant le processus de recherche de la transformation qui maximise la similarité entre l'image transformée et l'image consigne.

Donc on propose de voir les choses de la façon suivante :

- **Choix des solutions avec les métaheuristiques et les transformations possibles :** L'image à recaler et l'image de références avec quelques paramètres vont définir le problème de recalage à résoudre.
- **Sélection de la meilleure solution a l'aide des optimiseurs :** Pour ce problème on peut associer un optimisateur de type quelconque. Ce dernier doit permettre à la transformation qui maximisera la similarité entre l'image consigne et l'image à recaler transformée, définie dans le problème en question.
- **Application de la transformation choisie d'une solution sélectionnée:** Une fois la meilleure transformation choisie, on l'applique sur l'image à recaler pour obtenir une nouvelle image recalée dont la similarité avec l'image consigne est maximale.

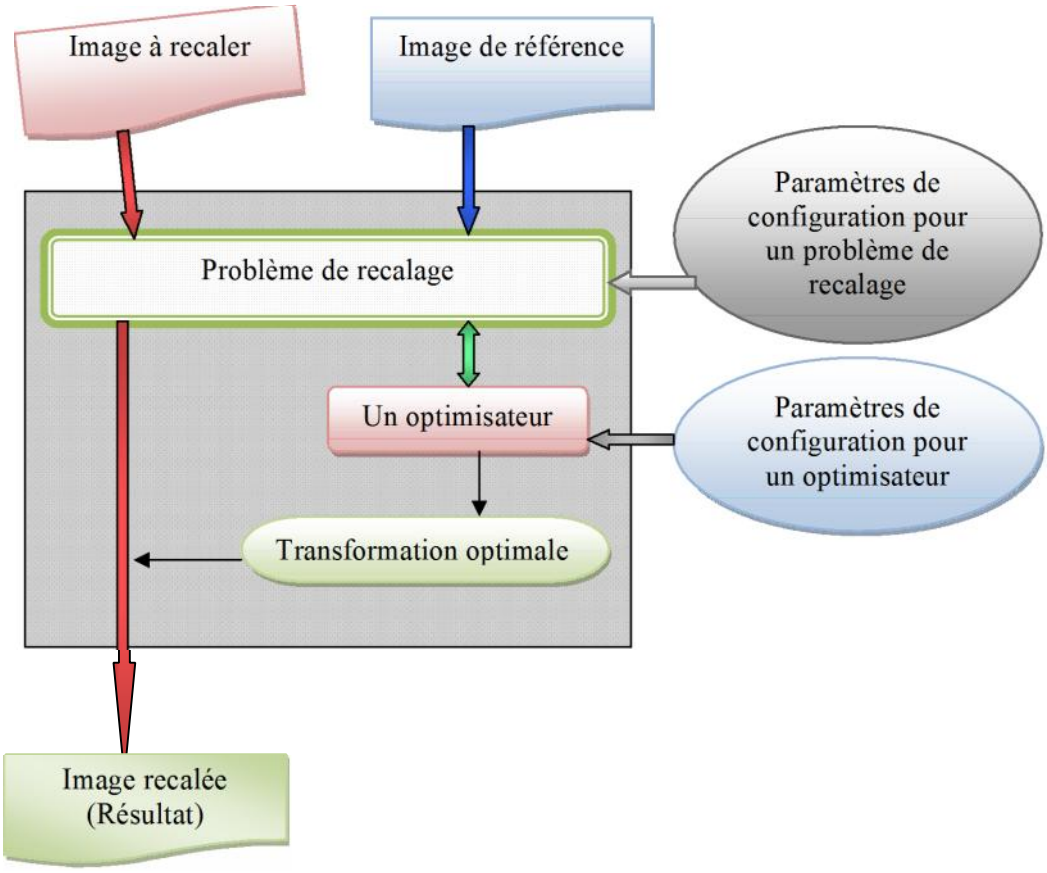


Figure 32. Schéma fonctionnel naturel d'un processus de recalage.

D'après la figure précédente, on constate que la transformation optimale est le seul lien qui relie l'optimisateur et le problème de recalage. L'optimisateur sera considéré comme étant une boite noire, qui a pour tâche la résolution du problème en entrée et la transformation optimale en sortie.

Figure 33. Entrées et sorties d'une méthode d'optimisation.

VI.2.1. Recalage par métaheuristiques

Il suffira de remplacer la boite noire « Optimisateur » par une métaheuristique appropriée pour obtenir une méthode de recalage basée sur métaheuristique.

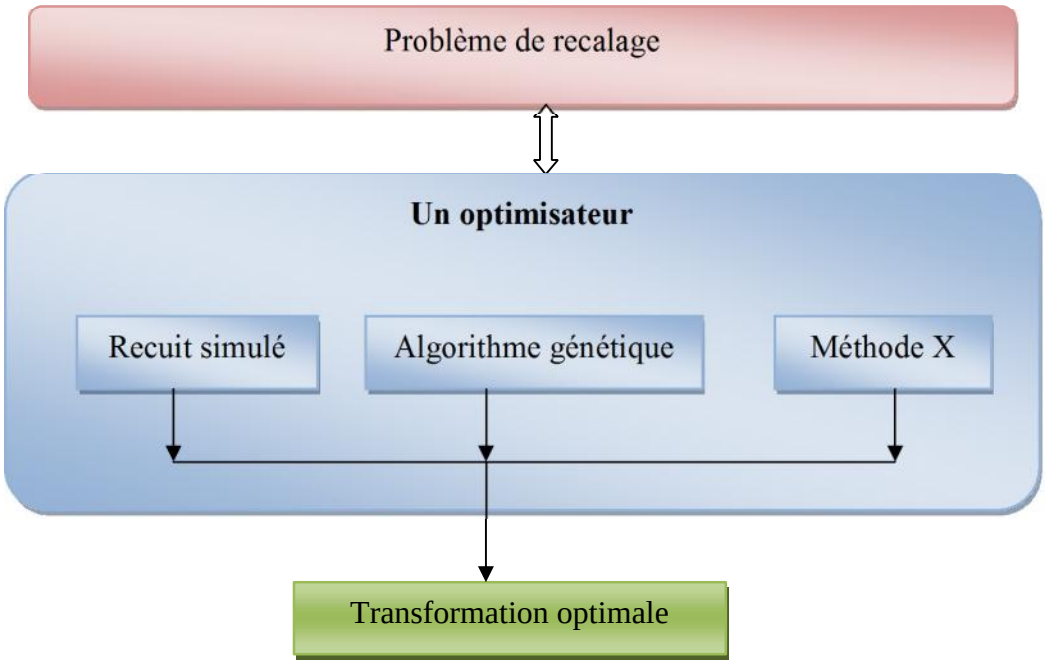


Figure 34. Recalage par métaheuristiques.

Dans ce qui précède, on a bien situé la place de la méthode d'optimisation dans le schéma de recalage, et on a réussi à isoler l'optimisateur du reste des parties du schéma, en ne gardant que les ports d'entrées et de sorties. Cela facilitera énormément d'aborder le problème, sa conception et sa mise en œuvre.

Nous présenterons ci-après les détails des schémas de recalage basés sur les métaheuristiques étudiés.

VI.2.2. Recalage par algorithmes génétiques

Il est à noter que nous avons opté de paramétrer le maximum des paramètres d'un algorithme génétique. Vous trouverez plusieurs choix pour le même paramètre. Cela peut enrichir l'étape de tests et des résultats.

Trois schémas de codages sont considérés naturels, binaire et quantique.

VI.2.2.1. Recalage par AGs en représentation naturelle

Pour aborder la conception du schéma de recalage à base d'algorithme génétique, il est nécessaire de définir tous les aspects génétiques pour le problème de recalage, et on va commencer par les individus

VI.2.2.2. Structure d'un individu

Un individu doit représenter une solution d'un problème d'optimisation, les solutions d'un problème de recalage ne sont autres que des transformations. Donc un individu dans un algorithme génétique résolvant un problème de recalage représentera une transformation.

Une transformation est caractérisée par un ensemble de variables, ces variables représentent les différents paramètres de la transformation, par exemple la transformation rigide sera représentée par trois variables, une pour modéliser la translation en axe des X (T1), une autre pour la translation en Y (T2) et la dernière pour l'angle de rotation (α).

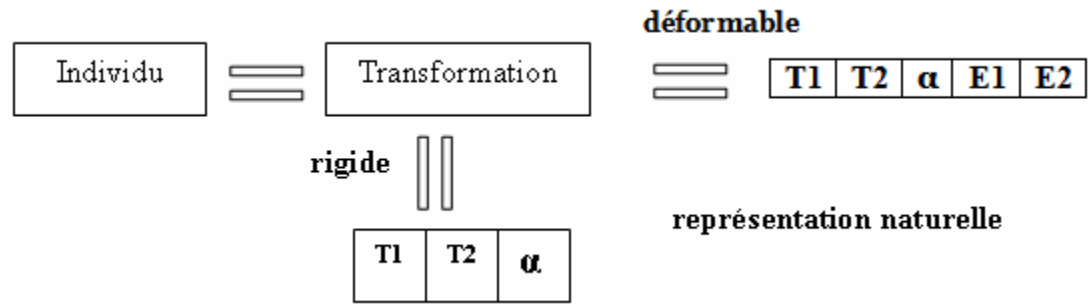


Figure 35. Représentation naturelle d'un individu (transformation).

Pour le déformable on ajoute l'élasticité en X (E1) et en Y (E2).

Adaptation d'un individu

La fitness d'un individu devra refléter le taux de similarité entre l'image consigne et l'image à recaler déformée par la transformation représentée par cet individu.

Dans le premier chapitre nous avons proposé six critères de similarité, rappelons les l'erreur quadratique, l'entropie, la covariance, le rapport de corrélation, l'information mutuelle et l'entropie mutuelle. Chaque critère de similarité va définir une fonction de fitness.

Voici le schéma générique d'une fonction d'adaptation d'un individu représentant une transformation rigide

Fitness (***Individu***)
Début
- Récupérer l'image consigne ***Ic*** et l'image à recaler ***Ir*** à partir du problème à résoudre.
- Récupérer ***T1*** et ***T2*** les translations en axe des X et en axe des Y, et ***Alpha*** l'angle de rotation à partir de l'individu ***Individu***.
- Calculer l'image transformée ***It*** = **TransformerR**(***Ir***, ***T1***, ***T2***, ***Alpha***).
- Selon le critère de similarité **Sim** retourner la valeur de **Sim**(***Ic***, ***It***).

Fin

Algorithme 2. Adaptation d'un individu.

La fonction **'TransformerR'** va retourner la transformation rigide (de paramètres ***T1***, ***T2***, ***Alpha***) de l'image ***Ir***, il faut dire que le centre de rotation sera le centre de gravité de l'image.

TransformerR (**Image_à_transformer, T1**, **T2**, **Alpha**)
Début
- Réserver de l'espace mémoire pour l'image **Image_Résultat**.
- soit (x0, y0) les coordonnées du centre de l'image **Image_à_transformer**.
- **Pour** chaque pixel (**lig**, **col**)
  - Calculer $\begin{pmatrix} lig' \\ col' \end{pmatrix} = \begin{bmatrix} \cos(alpha) & -\sin(alpha) \\ \sin(alpha) & \cos(alpha) \end{bmatrix} * \begin{pmatrix} lig - y0 \\ col - x0 \end{pmatrix} + \begin{pmatrix} T2 + y0 \\ T1 + x0 \end{pmatrix}$
  - Le pixel (**lig**, **col**) de l'image **Image_Résultat** aura le niveau de gris du pixel (**lig'**, **col'**) de l'image **Image_à_transformer**.
- **Fin pour**

Fin

Algorithme 3. Transformation rigide d'une image.

Dans le cas déformable on multiplie les grandeurs (lig' et col') par les valeurs de E1 et E2 respectivement.

La population initiale

Les individus de la population initiale sont générés aléatoirement dans un espace fixé par l'utilisateur. Ce choix garantira la diversité et l'occupation intelligente de l'espace de recherche, ce qui conduira à une meilleure exploration de l'espace de recherche.

Evaluation des individus d'une population

L'évaluation des individus d'une population donnée, consiste à calculer la fitness de chaque individu (si elle n'a pas été calculée préalablement).

Nous avons défini deux schémas d'évaluation, avec tri et sans tri. L'évaluation sans tri sera envisagée lors de l'utilisation des opérateurs génétiques (sélection et croisement) par tournois, celle avec tri sera utilisée avec les autres opérateurs génétiques.

Evaluer_population
Début
- Pour tout individu **Ind** de la population
  - Si (la fitness de **Ind** n'a pas été évaluée) alors ***calculer_fitness***(**Ind**)
- Fin pour

// si le tri est nécessaire alors les individus seront triés du moins adapté au plus adapté
Fin

Algorithme 4. Evaluation des individus d'une population.

Opérateurs de sélection génétique

Le but de cet opérateur est de choisir les individus à croiser, et de les mettre dans le bassin génétique.

Tous les schémas de sélection proposés et étudiés dans le (chapitre), peuvent être implémentés. On les classera en trois classes sélection élitiste, probabiliste et par tournois.

Sélection élitiste

Eliminer les individus dont le taux d'adaptation est inférieur

Sélection_élitiste
Début
- ***Tant que*** (le nombre de des individus > nombre des individus à sélectionner) ***faire***
  - Supprimer l'individu le moins adapté
  - ***Fin faire.***
- Mettre les individus sélectionnés dans le bassin génétique.

Fin

Algorithme 5. Sélection élitiste.

Sélection probabiliste

Dans ce genre de méthodes, un individu aura une probabilité de sélection ***Ps***,

Sélection probabiliste
Début
//L'évaluation était avec tri
- ***Tant que*** (le nombre de des individus sélectionnés < nombre des individus à sélectionner) ***faire***
  - Soit **Ind** l'individu le moins adapté de la population.
  - Si (uniforme [0,1] <= ***Ps***(**Ind**)) **alors**
    - Mettre **Ind** dans le bassin génétique
  - **Fin si**
- **Fin faire**

Fin

Algorithme 6. Sélection probabiliste.

Selon la méthode de calcul de la probabilité de sélection Ps, nous avons défini deux schémas de sélection probabiliste sélection par roue de la fortune et sélection par rang de classement.

Sélection par tournois binaires

Il s'agit de tirer arbitrairement deux individus de la population, garder l'individu le mieux adapté et éliminer l'autre.

Sélection par tournois binaires
Début
- ***Tant que*** (le nombre de des individus sélectionnés < nombre des individus à sélectionner) ***faire***
  - Soit **Ind1** et **Ind2** deux individus de la population courante.
  - Si (fitness(**Ind1**) < fitness(**Ind2**)) **alors**
    - Mettre **Ind2** dans le bassin génétique
  - Sinon
    - Mettre **Ind1** dans le bassin génétique
  - **Fin si**
- **Fin faire**

Fin

Algorithme 7. Sélection par tournois binaires.

Opérateurs de croisement génétique

Cet opérateur est appliqué selon une probabilité Pm, cette probabilité ainsi que les divers choix de l'opérateur sont fixés par l'utilisateur. On a fait la distinction entre deux aspects comment croiser les individus ? Et quels sont les individus (parmi les individus du bassin génétiques) à croiser ? La première question définit l'opérateur de croisement au niveau individus, la deuxième définit l'opérateur de croisement au niveau algorithme.

Croisement au niveau individus

On a adopté le schéma présenté dans la partie de l'état de l'art, deux parents à croiser pour engendrer deux enfants avec la probabilité Pc. Ce croisement garantira le partage efficace et intelligent des caractéristiques des parents.

Croiser (Parent1, Parent2, Enfant1, Enfant2, Pc)
Début
- **Pour** (chaque gène G) **faire**
  - **Si** (uniforme [0,1] <= Pc) **alors**
    - a←uniforme [0, 1]
    - Enfant1.G ← (a* Parent1.G) + ((1-a)*Parent2).
    - Enfant2.G ← (a* Parent2.G) + ((1-a)*Parent1).
  - **Sinon**
    - **Si** (uniforme [0, 1] <= 1/2) **alors**
      - Enfant1.G ← Parent1.G
      - Enfant2.G ← Parent2.G
    - **Sinon**
      - Enfant1.G ← Parent2.G
      - Enfant2.G ← Parent1.G
    - **Fin si**
  - **Fin si**
- **Fin faire**

Fin

Algorithme 8. Croisement des individus.

Opérateur de croisement au niveau algorithme

En réalité, il s'agit de préciser qui va être croisé avec qui ! On a proposé trois schémas distincts

- Croiser le meilleur individu avec son successeur.
- Croiser le meilleur avec le mauvais.
- Croisement par tournois binaires.

Croisement (***Pc***)
Début
- **Tant que** (le bassin n'est pas vide) **faire**
  - **Si** (il reste plus que deux individus dans le bassin génétique) **alors**
    - Sélectionner deux individus **P1** et **P2** selon un critère particulier
    - Croiser (**P1**, **P2**, **E1**, **E2**, ***Pc***)
    - Réintégrer **P1**, **P2**, **E1**, **E2** dans la population
    - Supprimer **P1**, **P2** dans le bassin
  - **Sinon** (il ne reste qu'un seul individu dans le bassin) **alors**
    - Muter l'individu
    - L'intégrer dans la population.
  - **Fin si**
- **Fin faire**

Fin

Algorithme 9. Opérateur de croisement (quels sont les individus à croiser).

Opérateur de mutation

Dans un algorithme génétique, cet opérateur est appliqué selon une probabilité **Pm**, la valeur de probabilité et fixé par l'utilisateur

Mutation (***Pm***)
Début
- **Si** (uniforme [0, 1] <= ***Pm***) **alors**
  - Calculer le nombre d'individus à muter =***Pm**** nombre total des individus
  - Choisir les individus à muter
  - **Pour** (chaque individu choisi) **faire**
    - Appliquer la méthode de mutation adoptée
    - Le réintégrer dans la population
  - **Fin faire**
- **Fin si**

Fin

Algorithme 10. Opérateur de mutation.

Le choix des individus à muter se fait arbitrairement. Tandis que les méthodes de mutations proposées sont

- Génération d'un nouvel individu.
- Ajout d'un vecteur d'éléments aléatoires.
- L'opérateur de mutation définit dans les stratégies d'évolutions [**SAL01**].

Les deux premières sont évidentes, alors que la troisième méthode consiste à calculer la variance de chaque gène, et la mutation soit

Muter (**Ind**)
Début
- **Pour** (chaque gène **G**) **faire**
  - Récupérer la variance **V.G** relative à **G** de tous les individus de la population courante.
  - **Ind**.G ← **Ind**.G + Normal (0, **V.G**)
- **Fin faire**

Fin

Algorithme 11. Opérateur de mutation de la méthode stratégies d'évolution.

Critère de fin de recherche

Le critère de fin de recherche utiliser est l'homogénéité de la population, la décision se fait par le calcul de la variance de chaque gène (le calcul se fait durant l'étape de l'évaluation). On dira que la population devient homogène si la variance des déplacements devient inférieur à 0.5 pixel, et celui de rotation de 0.05 radian.

La différence maximale entre deux paramètres de translation de deux individus d'une population homogène de dépassera jamais 1 pixel, celles entre rotations n'excèdera pas 0.1 radian.

D'atteindre l'homogénéité durant un processus d'optimisation nécessitera, peut être, plusieurs générations, cela peut entraîner un surcoût non négligeable. Donc, et pour contrôler la progression du processus d'optimisation, l'utilisateur peut fixer le nombre maximale de générations à évaluer, une fois atteint, l'algorithme s'achève, cela quelque soit le statut de l'homogénéité de la population courante.

Schéma général des algorithmes génétiques

Si on récapitule, on commence par l'initialisation qui consiste en une génération aléatoire des individus dans l'espace de recherche, puis dans le corps de l'algorithme, et durant une génération on doit évaluer la population, l'évaluation peut se faire avec un tri ou sans tri.

Si les individus sont triés alors on peut appliquer les opérateurs génétiques (sélection croisement et mutation) qui nécessitent le tri des individus, dans ce cas on peut appliquer les sélections par élitiste, par roue de fortune ou par rang de classement. Et on peut adopter les croisements meilleurs/meilleurs ou meilleurs/mauvais. Les tournois binaires, pour sélection et croisement, ne nécessitent pas le tri des individus, donc on peut les utiliser avec l'évaluation sans tri.

Si les individus, d'une population donnée, deviennent homogènes ou si l'on atteint le nombre maximal de génération alors l'algorithme prend fin.

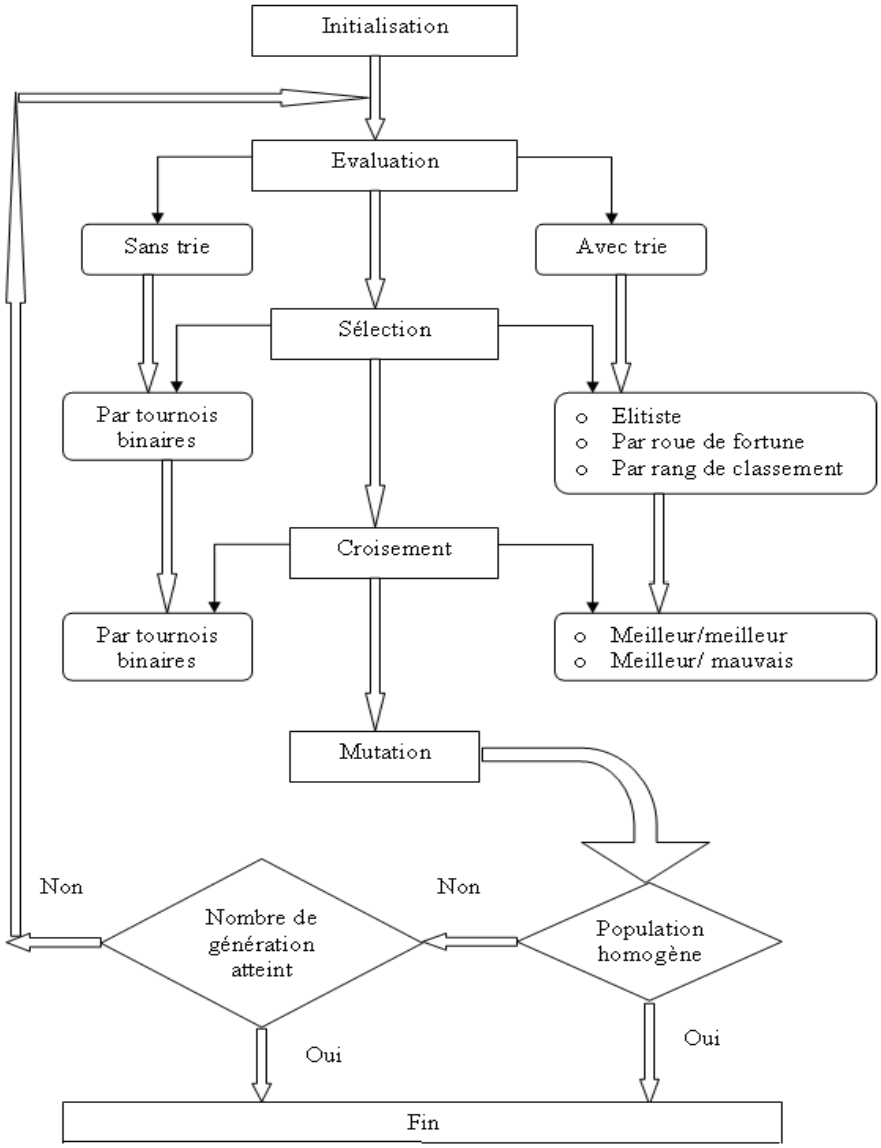


Figure 36. Schéma général (avec choix des opérateurs) de l'évolution génétique.

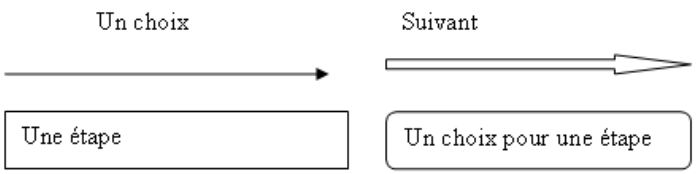


Figure 37. Etape et choix pour une étape.

VI.2.2.3. Recalage par AG binaires

Il y a quelques notions qui vont être changées, et cela sera surtout au niveau définition d'un individu, et opérateurs génétiques au niveau individu. La majorité des notions présentées à travers le chapitre précédent gardent les mêmes définitions du paragraphe (§VI.2.2.1).

Structure d'un individu

Cette fois ci, l'individu représentera une transformation codée en une chaîne binaire.

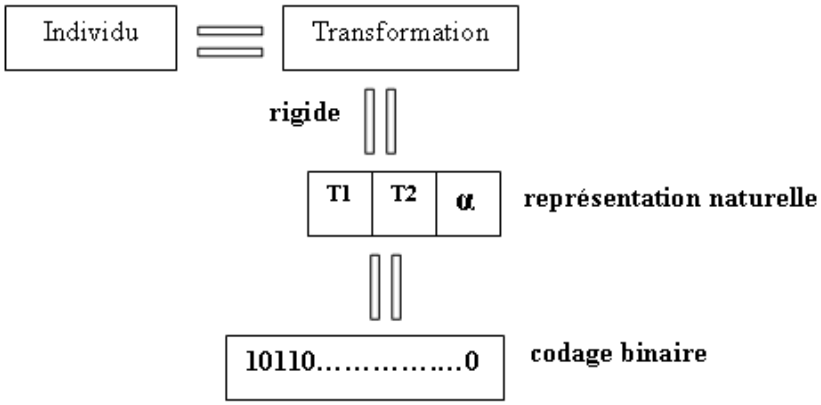


Figure 38. Individu en codage binaire.

La transformation codée sera trouvée par des concaténations des images codées de chacune de ses variables, cette concaténation se traduit par des multiplications et des additions.

Le nombre de Bits nécessaire pour codée une transformation dépendra de l'espace de recherche, ce dernier étant un ensemble des espaces de variations des paramètres. Le nombre de bits est évalué de la façon suivante

Nombre de bits nécessaires
Début
- ***Produit*** ← 1
- **Pour** (tout espace de variation d'un variable) **faire**
  - Récupérer la distance ***D*** entre les bornes de l'espace
  - ***Produit*** ← ***produit*** * ***D***
- **Fin faire**
- Retourner la valeur logarithmique à base de 2 de ***produit*** = log (2, ***produit***).

Fin

Algorithme 12. Nombre de bits nécessaire pour coder les solutions d'un espace de recherche.

Une fonction de décodage est nécessaire pour garantir l'accès aux paramètres de la transformation, cette fonction fonctionnera de la façon suivante

Décodage
Début
- Convertir la chaîne binaire en un nombre entier ***E***.
- Pour chaque espace de variation d'une variable de la transformation
  - Récupérer la borne inférieure ***Inf***, supérieure ***Sup*** ainsi que la distance entre les deux ***D***=***Sup***-**Inf**.
  - La variable décodée ← ((E mod ***D***) – ***Inf***);
  - ***E*** ← ***E*** / ***D*** ;
- **Fin faire**

Fin

Algorithme 13. Décodage des chaînes binaires.

La fonction de codage est pratiquement non utilisée.

Adaptation d'un individu

Avant de calculer la fitness d'un individu il faut tout d'abord le décoder, une fois décodé, on suit les mêmes démarches d'évaluation définies dans le schéma de codage non binaire. c.à.d calculer l'image transformée, et retourner la valeur de similarité entre elle et l'image consigne.

Croisement au niveau individus

Croiser (Parent1, Parent2, Enfant1, Enfant2, points)
Début
- Découper (versionBinaire (Parent1), points, BlocsParent1)
- Découper (versionBinaire (Parent2), points, BlocsParent2)
- Tour ← vrai
- **Pour** (chaque bloc B) **faire**
  - **Si** (Tour = vrai) **alors**
    - BlocsEnfant1 [B] ← BlocsParent1 [B]
    - BlocsEnfant2 [B]← BlocsParent2 [B]
  - **Sinon**
    - BlocsEnfant2 [B] ← BlocsParent1 [B]
    - BlocsEnfant1 [B] ← BlocsParent2 [B]
  - Tour ← non Tour
  - **Fin si**
- **Fin faire**
- ConstruireChaine (BlocsEnfant1, versionBinaire (Enfant1))
- ConstruireChaine (BlocsEnfant2, versionBinaire (Enfant2))

Fin

Algorithme 14. Croisement par échange des blocs binaires.

Le croisement binaire, se fait par l'échange de blocs (sous chaînes) binaires, ces blocs sont délimités par des points. On propose deux méthodes pour générer ces points la première génère des points équidistants, la deuxième génère des points aléatoires (suivant la loi uniforme).

La procédure ***Découper***, permet de découper une chaîne binaire quantique en un ensemble de blocs, les points de coupures sont passés en paramètre. A l'inverse de ***ConstruireChaine***, qui concatène des blocs binaires pour obtenir une seule chaîne binaire.

Au niveau algorithme on peut toujours appliquer l'un des trois schémas de croisement au sens meilleurs/meilleurs, meilleurs/mauvais ou encore par tournois binaires.

Opérateur de mutation

Dans un algorithme génétique, cet opérateur est appliqué selon une probabilité Pm, la valeur de probabilité est fixée par l'utilisateur

Mutation (***Pm***)
Début
- **Si** (uniforme [0, 1] <= ***Pm***) **alors**
  - Calculer le nombre d'individus à muter =***Pm**** nombre total des individus
  - Choisir les individus à muter
  - **Pour** (chaque individu choisi) **faire**
    - Appliquer la méthode de mutation adoptée
    - Le réintégrer dans la population
  - **Fin faire**
- **Fin si**

Fin

Algorithme 15. Mutation par inversion de bits binaires.

Schéma général d'optimisation avec les AG binaires

Le schéma général est le même que celui d'un algorithme génétique en représentation naturelle, avec l'application des opérateurs de reproduction binaires au lieu des opérateurs non binaires.

Recalage par AG binaires
Début
- Générer la population initiale aléatoirement.
- **Tant que** non fin **faire**
  - décoder les individus binaires pour obtenir les individus ordinaires.
  - Evaluer les individus ordinaires.
  - Sélectionner les individus binaires.
  - Evolution en utilisant les opérateurs d'évolution binaires
    - Appliquer l'opérateur de croisement binaire
    - Appliquer l'opérateur de mutation binaire.
  - **Si** (homogénéité de la population ordinaire ou le nombre de maximal de générations atteint) **alors** fin
- **Fin faire**

Fin

Algorithme 16. Evolution génétique binaire pour résoudre le problème de recalage.

Le test de l'homogénéité sera fait en utilisant la population d'individus ordinaires, de la façon décrite dans le paragraphe (*§0*).

VI.2.2.4. Recalage par algorithmes génétiques quantiques

Seuls les opérateurs définis au niveau individu qui vont être redéfinis, les autres restent inchangés.

Structure d'un individu

Cette fois ci, l'individu représentera une transformation codée en une chaîne binaire, la seule différence est au lieu d'utiliser des bits classiques, on utilise des bits quantiques.

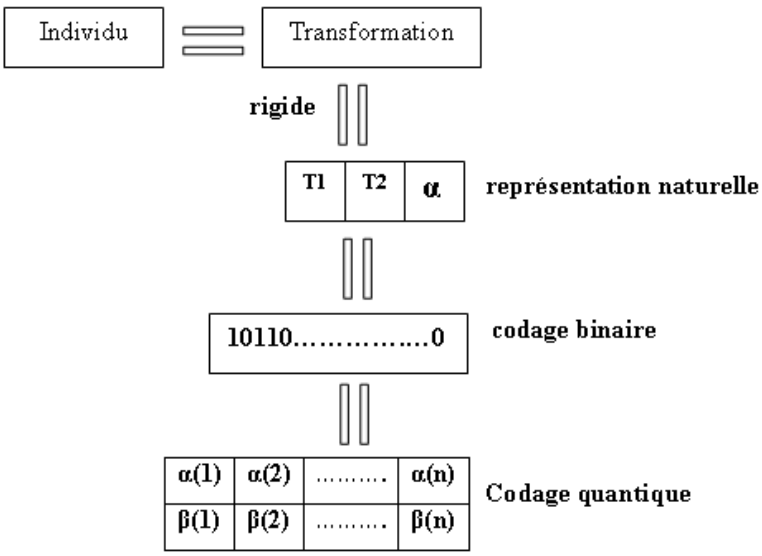


Figure 39. Individus quantiques.

Pour performer l'opération de décodage quantique on aura besoin d'un opérateur d'observation quantique, cet opérateur permet de récupérer la chaîne binaire classique à partir de la chaîne binaire quantique, elle fonctionne de la façon suivante :

Observation quantique (***chaine_quantique***, ***chaine_binaire***)
Début
- **Pour** (tout QuBit **QB** de la ***chaîne_quantique***) **faire**
  - **Si** $((\mathbf{QB}.\alpha)^2 > (\mathbf{QB}.\beta)^2)$ **alors**
    - Mettre à faux (0) le bit correspondant dans la ***chaine_binaire***.
  - **Sinon**
    - Le mettre à vrai (1).
  - **Fin si**
- **Fin faire**

Fin

Algorithme 17. Opérateur d'observation quantique.

Une fois l'image quantique de l'individu observée (on obtient ainsi la version binaire de la transformation), on peut faire appelle à la procédure de décodage binaire pour obtenir la représentation naturelle de l'individu.

Décodage quantique (***individu***)
Début
- Observation quantique (verionQuantique (***individu***), versionBinaire (***individu***))
- Décodage (versionBinaire (***individu***)).

Fin

Algorithme 18. Décodage d'un individu quantique.

Adaptation d'un individu

Avant de calculer la fitness d'un individu, il faut tout d'abord le décoder. Une fois décodé, on suit les mêmes démarches d'évaluation définies dans le schéma de codage non binaire. c.à.d calculer l'image transformée, et retourner la valeur de similarité entre elle et l'image consigne.

Opérateurs de croisement génétique

Qu-Croiser (Parent1, Parent2, Enfant1, Enfant2, points)
Début
- QuDécouper (versionQuantique(Parent1), points, QuBlocsParent1)
- QuDécouper (versionQuantique(Parent2), points, QuBlocsParent2)
- Tour ← vrai
- **Pour** (chaque Qubloc B) **faire**
  - **Si** (Tour = vrai) **alors**
    - QuBlocsEnfant1 [B] ← QuBlocsParent1 [B]
    - QuBlocsEnfant2 [B] ← QuBlocsParent2 [B]
  - **Sinon**
    - QuBlocsEnfant2 [B] ← QuBlocsParent1 [B]
    - QuBlocsEnfant1 [B] ← QuBlocsParent2 [B]
  - Tour ← non Tour
  - **Fin si**
- **Fin faire**
- QuConstruireChaine (QuBlocsEnfant1, versionQuantique(Enfant1))
- QuConstruireChaine (QuBlocsEnfant2, versionQuantique(Enfant2))

Fin

Algorithme 19. Croisement par échange de blocs quantiques.

Le croisement quantique, tout comme le croisement binaire, se fait par l'échange de blocs (sous chaînes) quantiques, ces blocs sont délimités par des points. On propose deux méthodes pour générer ces points la première génère des points équidistants, la deuxième génère des points aléatoires (suivant la loi uniforme).

La procédure QuDécouper, permet de découper une chaîne binaire quantique en un ensemble de blocs, les points de coupures sont passés en paramètre. A l'inverse de QuConstruireChaine, qui concatène des blocs quantiques pour obtenir une seule chaîne quantique.

Au niveau algorithme on peut toujours appliquer l'un des trois schémas de croisement au sens meilleurs/meilleurs, meilleurs/mauvais ou encore par tournois binaires

Opérateur de mutation

La mutation quantique est appliquée de sorte d'inverser des qubits, le nombre de qubits à inverser ainsi que la méthode de leur génération est fixés par l'utilisateur. La mutation quantique se déroule de la façon suivante

Qu-Muter (**Ind, points_à_inverser**)
Début
- **Pour** (chaque QuB dans la liste **points_à_inverser**) **faire**
  - Appliquer le circuit **NOT** quantique sur le qubit **QuB**.
- **Fin faire**

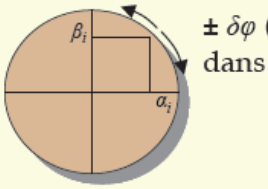
Fin

Algorithme 20. Mutation quantique d'un individu.

Rotation quantique [KUK02]

Etant un opérateur très important dans le schéma d'évolution par algorithmes génétiques quantiques. Elle est utilisée pour générer une nouvelle population quantique à partir d'une autre.

Pour un individu quantique, la rotation quantique consiste à fixer une chaîne binaire (généralement celle de l'individu le mieux adapté), ensuite appliquer le circuit de rotation quantique sur chaque qubit de l'individu, l'angle de rotation est fixée pour chaque qubit selon le tableau suivant



± $\delta\varphi$ (selon la valeur du bit correspondent dans la meilleure solution)

Angle	Valeur du bit de référence	β	α
+δθ	1	> 0	> 0
-δθ	0	> 0	> 0
-δθ	1	< 0	> 0
+δθ	0	< 0	> 0
-δθ	1	> 0	< 0
+δθ	0	> 0	< 0
+δθ	1	< 0	< 0
-δθ	0	< 0	< 0

Valeurs prises par l'angle de rotation

Tableau 5. Table de rotation qauntique.

La procédure de rotation quantique d'un individu d'une population est

Rotation quantique
Début
- Récupérer la chaîne de référence **Ref**←versionBinaire (meilleur individu quantique).
- **Pour** (chaque individu quantique **Q) faire**
  - **Pour** (chaque *i-ème* qubit **qb (α, β)** de l'individu **Q**) **faire**
    - Déterminer l'angle de rotation **δθ** en utilisant l'i-ème bit de **Ref** et les amplitudes de probabilités **(α, β).**
    - Calculer les nouvelles amplitudes **(α', β')** = rotation quantique **((α, β), δθ)**
    - Remplacer les amplitudes de probabilités du qubit **qb** par **(α', β').**
  - **Fin faire**
- **Fin faire**

Fin

Algorithme 21. Rotation quantique.

Schéma d'évolution génétique quantique pour le recalage

Recalage par AGQ
Début
- Générer la population quantique initiale aléatoirement.
- **Tant que** non fin **faire**
  - Observer les individus quantiques pour obtenir les individus ordinaires.
  - Evaluer les individus ordinaires.
  - Sélectionner les individus quantiques.
  - Evolution quantique des individus quantiques
    - Appliquer l'opérateur de croisement quantique
    - Appliquer l'opérateur de mutation quantique.
    - Appliquer la rotation quantique
  - **Si** (homogénéité de la population ordinaire ou le nombre de maximal de générations atteint) **alors** fin
- **Fin faire**

Fin

Algorithme 22. Evolution génétique quantique pour le recalage d'images.

Le test de l'homogénéité sera fait en utilisant la population d'individus ordinaires.

VI.2.3. Recalage par recuit simulé

Recuit simulé étant une méthode paramétrique, à plusieurs paramètres qui contrôlent l'efficacité de cette dernière. Dans notre cas le recalage des images, nous avons opté de paramétrer tous les atouts de recherche, pour but d'enrichir l'étape des tests et résultats.

Pour implémenter la méthode de recuit simulé, nous allons définir le squelette de se dernier. En première étape, on aborde la structure d'un état.

VI.2.3.1. Structure d'un état

Un état, dans recuit c'est la représentation physique des structures d'atomes en processus de cristallisation, l'état dans recuit simulé doit représenter une solution d'un problème d'optimisation, dans notre cas, une solution c'est une transformation d'image possible.

Une transformation est représentée par un vecteur de réels de taille (n), dans le cas d'une transformation rigide, le vecteur de transformation contient des translations en X et Y et une rotation Alpha. On rajoute les élasticités en X (E1) et en Y (E2) dans le cas déformable.

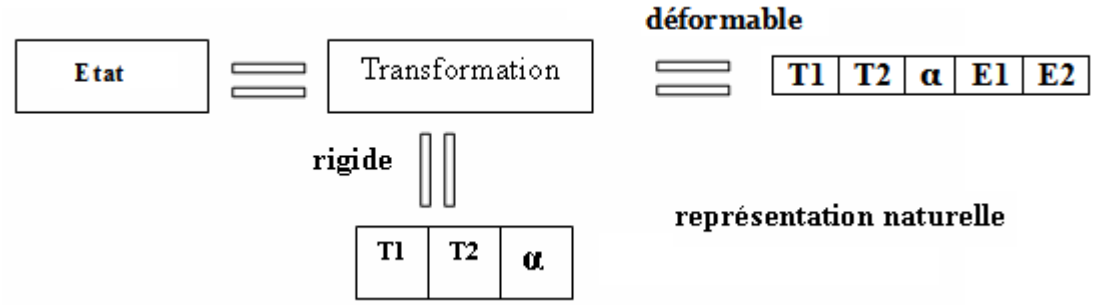


Figure 40. Structure d'un état.

VI.2.3.2. Energie d'un état

L'énergie d'un état reflète le taux de similarité entre deux images, l'image référence ou (consigne) et l'image à recaler transformée par la déformation représentée par l'état. Le calcul de l'énergie est le même que le calcul de fitness d'un individu.

VI.2.3.3. Initialisation

- **Etat initial**

Représente la transformation nulle, tous les paramètres sont initialisés à zéro.

- **Température initiale**

La température initiale est initialisée de telle sorte que la plus mauvaise transformation soit admise avec une probabilité de 80%.

L'initialisation automatique se déroule de la façon suivante

Initialisation_température(nombre_itérations_initialisé)
Début
- Générer un échantillon d'états aléatoirement et stocker leurs énergies
- Mettre l'énergie minimale dans Min.
- Mettre l'énergie maximale dans Max.
- Température_intiale= $-\frac{|Max - Min|}{Log(0.8)}$

Fin

Algorithme 23. Initialisation de la température

On peut aussi initialiser la température manuellement.

VI.2.3.4. Génération des états voisins

Nous avons implémenté plusieurs méthodes (ou manières) pour obtenir un nouvel état voisin à partir d'un autre état.

Les méthodes sont

- **Par mutation** elle consiste à spécifier aléatoirement des éléments dans le vecteur d'état courant, et changer ces derniers par des valeurs générées aléatoirement dans l'espace de recherche.
- **Par addition d'un vecteur aléatoire** l'état voisin=l'état courant+vecteur aléatoire.
- **Par changement périodique d'un seul élément** elle consiste à sélectionner un élément, et de lui affecter une valeur aléatoire dans l'espace de recherche. Les éléments seront sélectionnés au tour de rôle.
- **Générer indépendamment** elle consiste en une génération aléatoire d'un état indépendamment de l'état courant.

VI.2.3.5. Acceptation d'un état

L'admission de l'état voisin **X**, d'un état **Y** à la température **Tcourante** se fait selon le critère de métropolies, de la façon suivante

Acceptation (X, Y, Tcourante)
Début
- Si (uniforme [0, 1] <= $EXP\left(\frac{énergie(X) - énergie(Y)}{Tcourante}\right)$) alors
  - Accepter l'état voisin X
- Sinon
  - Ne pas accepter l'état X

Fin

Algorithme 24. Acceptation des états.

VI.2.3.6. Refroidissement de température

Le refroidissement de température, guide le recuit simulé a creusé, encore plus pour améliorer l'état optimal, différents types de refroidissement ont été exposés lors d'état de l'art, nous avons implémenté deux types

- **Refroidissement linéaire**

  TCourante=TCourante*Alpha, Alpha constante de refroidissement.

- **Refroidissement logarithmique**

  TCourante=TCourante/ln (numéro de l'étape en cours).

VI.2.3.7. Critère de fin de recherche

On propose les critères suivants

- **Un nombre maximal d'étapes**

```
Critère_de_fin()
Début
Si (nombre_étape_actuelle > nombre_max_étape) alors
    • Arrêter recherche.
Fin Si
Fin
```

Algorithme 25. Fin de recherche avec un nombre maximal d'étapes

- **Un pourcentage minimal de température**

```
Critère_de_fin()
Début
Si ((TCourante/TInitiale)< PourcentageFixe) alors
    • Arrêter recherche.
Fin Si
Fin
```

Algorithme 26. Fin de recherche avec une température minimale.

- **Un pourcentage minimal d'états acceptés**

```
Critère_de_fin()
Début
Si((Nb_états_accéptés/Nb_itérations_par_étape)< PourcentageFixe) alors
• Arrêter recherche.
Fin Si
Fin
```

Algorithme 27. Fin de recherche avec le taux des états acceptés

- **Un taux de similarité satisfaisant**

```
Critère_de_fin()
Début
Si (énergie (état_optimal))<= TauxFixe) alors
        • Arrêter recherche.
Fin Si
Fin
```

Algorithme 28. Fin de recherche avec une qualité satisfaisante.

VI.2.3.8. Le schéma général de recuit simulé

Tout en récapitulant les différentes étapes du parcours de la méthode, on va construire le schéma général de recuit simulé. Ce dernier commence par une étape d'initialisation (initialisation des états, initialisation de la température). Ensuite, il commence à explorer l'espace de recherche en générant de nouveaux états avec les méthodes de générations présentées, l'acceptation de ces états se fait selon le critère de métropolis.

A la fin de chaque étape le processus refroidit la température selon l'un des schémas proposés (linéaire ou logarithmique). Selon la température courante, on restreint l'espace de recherche sur le voisinage de la meilleure solution. Ensuite, on teste la fin de recherche suivant un des critères proposés? Pour que le processus d'optimisation prend fin.

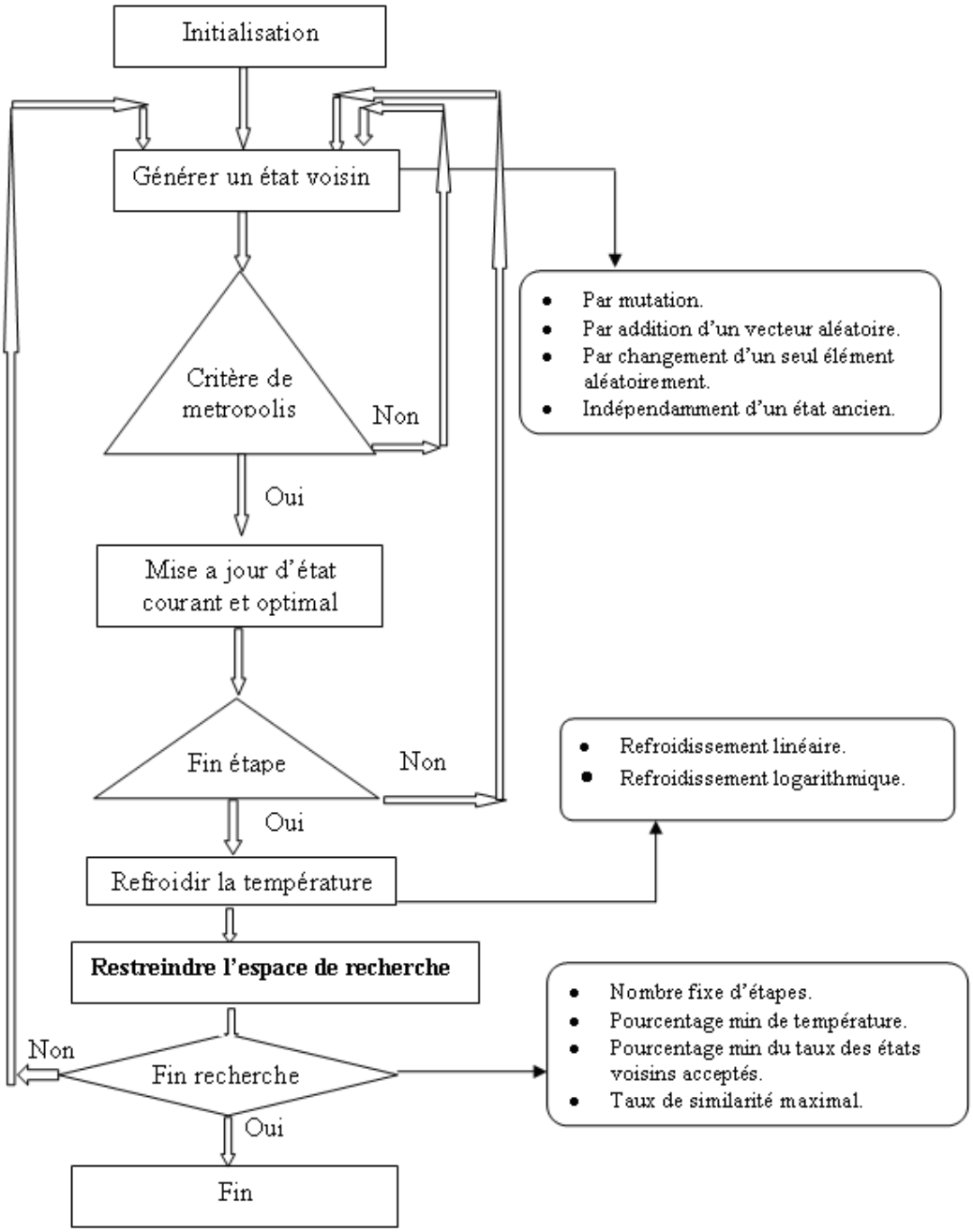


Figure 41. Optimisation par recuit simulé.

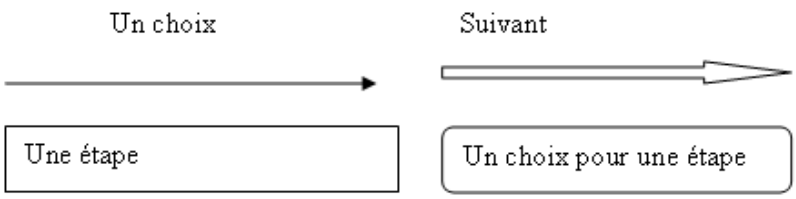


Figure 42. Clé pour le recuit simulé.

VI.2.4. Recalage par recuit simulé (en codage binaire)

Deux modifications sont envisagées pour la méthode:

VI.2.4.1. La structure d'un état

L'état est codifié en une chaîne binaire qui représente une solution (une transformation) après décodage.

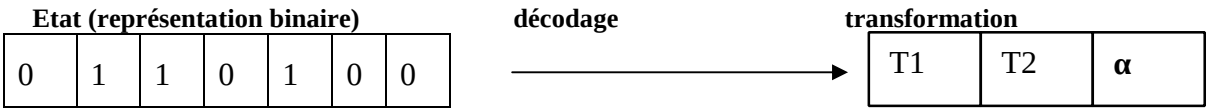


Figure 43. Etat en codage binaire.

VI.2.4.2. La génération de l'état voisin

Pour la génération d'un nouvel état, on va exercer une succession de changement de bits sur la chaîne binaire de l'état actuel et mettre le résultat dans l'état voisin. La sélection des bits à changer est aléatoire.

VI.3. CONCLUSION

Durant cette partie, intitulée conception nous avons présenté les principales méthodes d'optimisation avec leurs schémas liés au codage des solutions. Nous avons aussi, proposé un modèle de recalage à base des métaheuristiques, ce qui démontre l'adéquation théorique de ces derniers au problème de recalage.

CHAPITRE VII. MISE EN ŒUVRE

VII.1. INTRODUCTION

Pour la mise en œuvre on a réalisé un logiciel nommé « RIS-MH » : Recalage d'Images Satellitaires, une approche basée sur Méta Heuristiques, on a utilisé le compilateur C++ de Borland C++ Builder 6. On va tirer profit de la modélisation objet qui facilitera la gestion efficace des différentes entités, et de la puissance de calcul d'un compilateur C++.

VII.2. DEVELOPPEMENT DU LOGICIEL «RIS-MH»

«RIS-MH» est une plateforme puissante qui a été développé pour

- réaliser la tâche de recalage en utilisant des métaheuristiques
- Valider expérimentalement l'adéquation des métaheuristiques pour résoudre le problème de recalage
- Etudier et analyser les comportements de chaque métaheuristique, (qualité de recalage temps d'exécution totale, coût de chaque partie d'algorithme, …etc.).

Avec sa conception hautement flexible, grâce à la séparation sémantique des modules qui sont considérés comme étant des boites noires avec des entrées et des sorties, «RIS-MH» est toujours prête à recevoir des éventuelles modifications amélioratrices (voir ci-dessous).

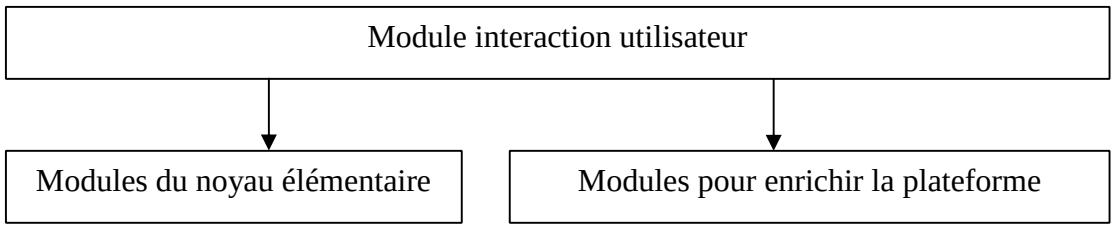


Figure 44. Séparation (sémantique) des modules de la plateforme «RIS-MH».

La plateforme est divisée en deux grandes parties la première partie comportera les modules communes et indispensables pour bien mener la tâche requise, elle ne peut rien pour résoudre le problème de recalage, la deuxième contient des modules qui, en utilisant les modules du noyau élémentaire, réaliseront efficacement la tâche requise.

VII.3. COMPILATEUR UTILISE

Le choix du compilateur c'est un facteur important puisque il a un impacte très important sur le coût calculatoire globale ainsi que la modélisation des objets qui facilitera la gestion des différentes entités du programme. Suite à ces réquisitions, on a analysé divers compilateurs (langages de programmation), pour le but de faire un choix optimal sur les deux contraintes temps de calcul et flexibilité du langage, les compilateurs éluent Java, Visuel Studio 2005 et .net (C++, C#, VB), Borland C++ et Borland Delphi.

Tableau 6. Tableau comparatif des compilateurs utilisés

Langages	Portabilité	Temps d'exécution	Flexibilité du langage
JAVA	+	-	+/-
VISUAL Studio.net	+	-	+
Borland Delphi	-	+	+/-
Borland C++	-	+	+

Analyse :

La portabilité dans notre réalisation n'est pas un facteur assez important, puisque l'application qu'on développe ne s'occupe pas de ce problème, par contre du coté d'optimisation et d'amélioration des résultats de recherche c'est les deux derniers facteurs font l'objet de distinctions et de choix du compilateur puisque ils jouent sur la rapidité d'exécution et la flexibilité du langage en matière de modélisation des solutions, et dans ces conditions là, le compilateur Borland C++ se distingue avec sa programmation orienté objet et facilitera la gestion efficace des différentes entités, de plus sa puissance de calcul est indiscutable.

VII.4. NOYAU ELEMENTAIRE

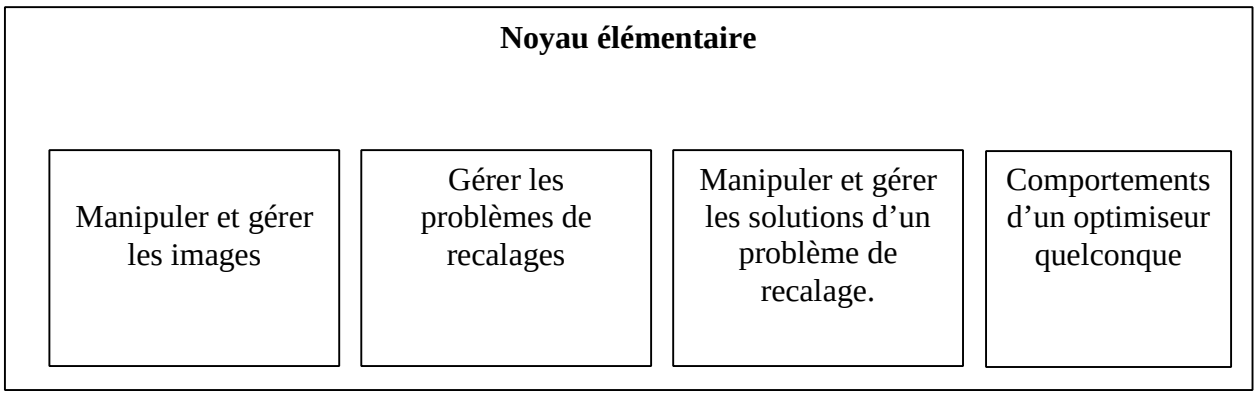


Figure 45. Les modules principaux du noyau élémentaire «RIS-MH».

VII.4.1.1. Classes et objets de l'Application

Cette partie contient les classes suivantes (*voir ci-dessus*):

- **CImage** pour la manipulation d'images en niveau de gris.
- **CRegistrationProblem** pour modéliser et gérer les problèmes de recalages.
- **CSolutionForOptimizer** modélise une solution d'un problème de recalage dans pour une méthode d'optimisation quelconque.
- **COptimizer** une classe de base qui modélise le comportement d'une méthode d'optimisation quelconque.

On va, à présent, détailler les principales classes du noyau, ainsi que leurs fonctions.

VII.4.1.2. La classe « CImage »

Entité permettant une modélisation simple des images en niveaux de gris, elle contient les informations suivantes

- Résolution de l'image
- Une matrice d'entiers pour stocker les valeurs des niveaux de gris des pixels.

Cette classe réalisera les tâches indispensables au bon déroulement du processus de recalage **les tests de similarité** et **les transformations d'images**.

VII.4.1.3. La classe « COptimizer »

C'est une classe de base pour toutes les méthodes d'optimisation, elle comporte les informations suivantes

- Un pointeur vers le problème de recalage à résoudre.
- L'espace de recherche.
- Le codage utilisé (naturel, binaire, quantique)
- La meilleure transformation rencontrée par l'optimisateur.
- Un pointeur vers le **thread** qui exécute le processus d'optimisation.
- Un pointeur vers la fenêtre qui contrôle la progression du processus d'optimisation.

La méthode la plus importante de cette classe est celle dédiée à l'optimisation Optimize(), cette méthode est virtuelle, donc toutes les méthodes d'optimisations qui vont hériter les propriétés de cette classe devront redéfinir cette méthode.

Optimize() s'exécute en thread, ce dernier est contrôlé par l'utilisateur par une fenêtre. Le thread et la fenêtre sont pointés par les membres cités précédemment.

VII.4.1.4. La classe « CRegistrationProblem »

Classe pour modéliser un problème de recalage. Une instance de cette classe contient les informations suivantes

- L'image consigne
- L'image à recaler
- Le critère de similarité utilisé pour exprimer la similarité entre les images
- Un pointeur vers l'instance de l'objet chargé de la tâche d'optimisation.

Un ***CRegistrationProblem*** évalue une transformation rigide, à travers la méthode ***GetCost(transformation),*** cette dernière fonctionne de la façon suivante

```
GetCost (transformation)
{
Image_transformée=Calculer_l'image_transformée(image_à_recaler, transformation) ;
Taux=similarité (image_transformée, image_consigne) ;
Coût=adapter (taux);
Retourner coût.
}
```

Algorithme 29. Dans «RIS-MH» c'est le problème à résoudre qui évalue une solution.

Toutes les méthodes d'optimisations ont été conçues dans un sens de minimisation, la fonction adapter permet de corriger l'information portée par le taux de similarité selon le critère de similarité utilisée de la façon suivante

Critère de similarité	Espace de variation	Ressemblance parfaite	Fonction d'adaptation
Rapport de corrélation	[0, 1]	Maximal	Coût= -taux
Covariance	IR	Maximale en valeur absolue	Coût= -|taux|
Entropie	IR+	Minimale	Coût= taux
Erreur quadratique	IR+	Minimale	Coût= taux
Entropie mutuelle	IR+	Minimale	Coût= taux
Information mutuelle	IR+	Maximale	Coût= -taux.

Tableau 7. Adapter les taux de similarités dans un context de minimisation.

VII.4.1.5. La classe « CSolutionForOptimizer »

Elle est à la fois

- **CNaturalSolution** représentation naturelle d'une solution.
- **CBinaryCodedSolution** codage binaire d'une solution.
- **CQuantumCodedSolution** codage quantique d'une solution.

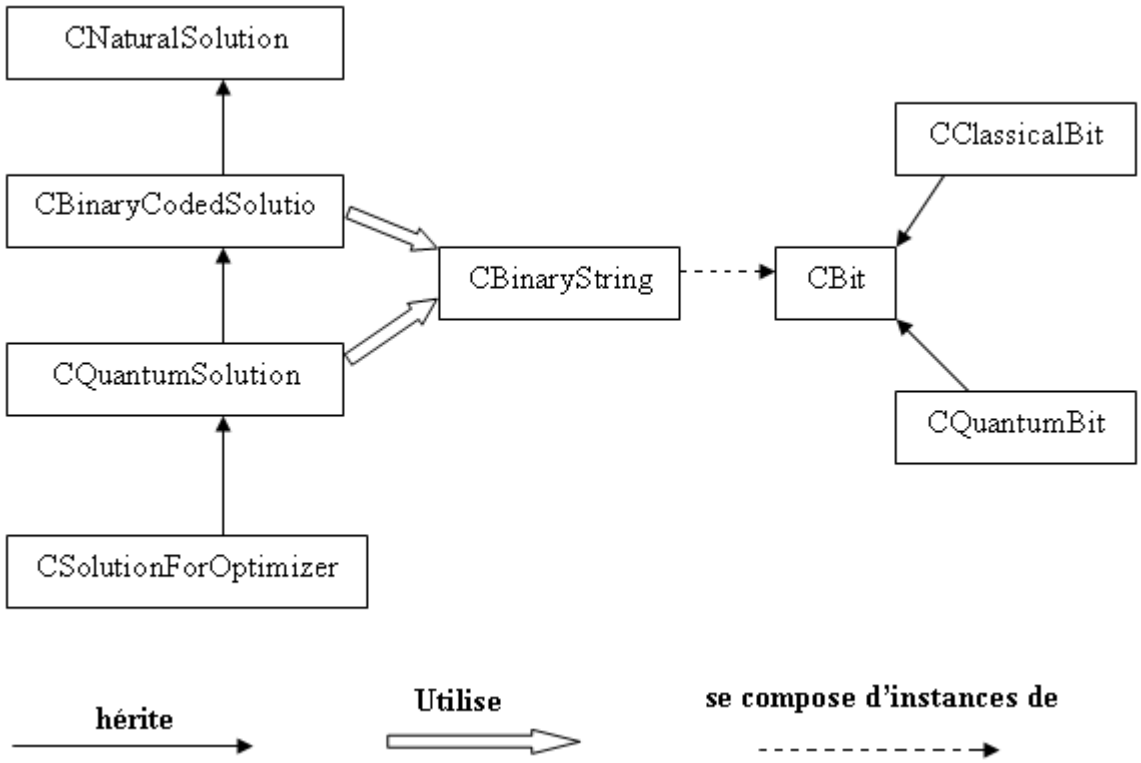


Figure 46. Graphe d'héritage, une solution peut etre binaire quantique ou encore naturelle.

Dans ce qui suit on détaillera les classes, de la plus élémentaire au plus complexe

CBit, CClassicalBit et CQuantumBit

Ils sont tous des bits, dont les données sont

- **CBit** pas de données, que des fonctions virtuelles qui ne font rien.
- **CClassicalBit** contient un booléen
- **CQuantumBit** contient deux réels, Alpha et Beta.

On peut donc implémenter les tâches virtuelles suivantes

- **Test ()** tester si le bit est à 1
  - Bit classique retourner la valeur du booléen
  - QuBit retourner la valeur booléenne de l'expression ($Alpha^2<Beta^2$).
- **Set ()** mettre le bit à 1
  - Bit classique mettre le booléen à vrai
  - quBit Alpha ← 0, et Beta ← 1.
- **Reset ()** mettre le bit à 0
  - Bit classique mettre le booléen à faux
  - QuBit Alpha ← 1, Beta ← 0
- **Flip ()** inverser le bit
  - Bit classique booléen ← Non (booléen)
  - QuBit inverser Alpha et Beta

- **Rotate (alpha)** rotation d'un bit
    - Bit classique non définie
    - QuBit performer la rotation quantique d'angle passée en paramètre.

Classe « CBinaryString »

Cette classe contient un vecteur de Bits (classiques ou quantiques), et un booléen pour indiquer le genre de bits utilisés. Elle accorde les méthodes suivantes

- **UseQuBits()** (resp. **UseClBits()**) la chaîne devient une chaîne de bits quantiques (resp. chaîne de bits classiques).
- **GetFromInt(nb, nb_bits)** crée une chaîne de nb_bits bits (leurs genre est indiqué par la variable membre), et qui sera l'image codée de l'entier nb.
- **Copy(CBinaryString* ch)** cette méthode permet d'avoir une copie intégrale (genre de bits, leurs paramètres, …etc.) de la chaine ch.
- **GetFromAnotherBS(CBinaryString* ch)** cette méthode permet d'avoir une copie des valeurs logiques des bits de la chaîne ch. On peut utiliser cette méthode pour avoir l'image binaire d'une chaîne quantique, ce qui facilitera l'implémentation de l'opérateur de d'observation quantique par la suite.

La classe « CNaturalSolution »

Contient un vecteur de réels, qui représentent les paramètres d'une transformation, elle contient aussi un pointeur vers l'espace de recherche. Elle accorde des méthodes de perturbation des valeurs des paramètres de la transformation représentée.

La classe « CBinaryCodedSolution »

Hérite de la classe CNaturalCodedSolution donc de toutes ses propriétés, y compris le vecteur des paramètres de la transformation. Elle ajoute une chaîne de bits classique, dont la taille est calculée en fonction de l'espace de recherche.

Les méthodes les plus importantes qu'elle offre sont

- **Decodate()** elle permet de décoder la chaîne binaire pour obtenir les valeurs des paramètres de la transformation et de les stocker dans le vecteur des réels, le décodage utilise l'espace de recherche.
- **FlipBits (liste)** elle permet d'inverser les bits dont le rang est dans la liste passée en paramètre, une opération de décodage est appelée à la fin de l'opération.

La classe « CQuantumCodedSolution »

Hérite de la classe CBinaryCodedSoluttion, elle ajoute une chaîne de bits quantiques dont la taille est évaluée de la même façon que la taille de la chaîne binaire.

Les méthodes les plus importantes sont

- **QuDecodate()** elle permet de décoder la chaîne quantique pour obtenir les bonnes valeurs des variables de la transformation, elle fonctionne en deux étapes

- Mesurer les états des **QuBits** de la chaîne quantique en utilisant la méthode **GetFromAnotherBS** de la chaîne binaire de la classe **CBinaryCodedSolution**. A la fin, la chaîne binaire contiendra les mêmes informations que la chaîne quantique.
- Appeler la méthode **Decodate**(), de la classe **CBinaryCodedSolution**.

- **FlipsQuBits(liste)** elle réalise une tâche similaire à celle **FlipBits()** de la classe CBinaryCodedSolution, sauf que cette fois elle opère la chaîne quantique. Une opération de décodage quantique est envisagée à la fin de cette méthode.

La classe « CSolutionForOptimizer »

Avant de détailler les membres de cette classe, on aime préciser qu'à tout moment les informations présentes dans les trois niveaux d'une solution (niveau naturel, binaire, quantique) soient cohérentes car toute perturbation d'une information d'un niveau donnée sera suivie par une opération de décodage adéquate.

Cette classe modélise une solution pour un problème de recalage, elle contient les informations suivantes

- Un booléen qui indique si la solution a été évaluée ou non.
- Un réel qui porte la qualité de la solution, si elle a été évaluée pour un problème de recalage.
- Un entier qui précise le niveau est la solution
  - **0** la solution utilise la représentation naturelle.
  - **1** la solution utilise le codage binaire.
  - **2** elle utilise le codage quantique.

Les méthodes les plus importantes sont

- **CreateSolution(COptimizer*)** la création consiste à
  - une génération aléatoire des paramètres de la transformation rigide
  - le genre de codage est celui de l'optimiseur
  - mettre à jours le pointeur de l'espace de recherche vers celui de l'optimiseur.
- **Evaluate(CRegistrationProblem* Pb)** l'évaluation se déroule de la façon suivante
  - Si la solution à été évaluée alors retourner sa qualité
  - Sinon alors appeler la méthode **GetCost()** (qui aura comme paramètre un pointeur vers le vecteur des variables de la transformation) du problème **Pb**. Et mettre à jours les variables concernées

VII.4.1.6. Remarques

Cette partie est nommée noyau élémentaire, contient toutes les tâches communes à tous les schémas possibles d'un processus de recalage basé sur métaheuristiques. Nous avons séparé la tâche d'optimisation, en fait, nous l'avons considéré comme étant une boite noire dont la seule sortie utile soit la meilleure solution pour un problème donné.

Cette façon de modéliser le problème augmentera la flexibilité de la plateforme, dans le sens où par la suite on peut rajouter n'importe quel type d'optimisateurs. Elle augmentera également l'utilité de nos bibliothèques des méthodes d'optimisations, car elles ne seront pas dédiées à un problème d'optimisation particulier mais à n'importe quel problème d'optimisation. Cette caractéristique nous a été accordée grâce à la stratégie d'évaluation des solutions, en fait c'est le problème qui les évalue et non pas les méthodes d'optimisation.

VII.5. MISE EN ŒUVRE DES METAHEURISTIQUES

Le noyau élémentaire comme son l'indique, n'est qu'une plateforme bien conçue pour supporter de différents types de méthodes d'optimisation, ainsi elle ne peut rien faire pour résoudre notre problème.

En effet, pour ajouter des bibliothèques pour des vrais méthodes d'optimisation il faut agir sur deux fronts, la classe qui modélise le comportement de l'optimisateur en question et qui doit hériter de la classe COptimizer, et éventuellement d'ajouter une classe pour modéliser les traitements supplémentaires qu'on peut faire sur une solution, dans ce cas la nouvelle classe doit hériter le comportement de la classe CSolutionForOptimizer.

VII.5.1. Mise en œuvre des méthodes « algorithmes génétiques »

VII.5.1.1. La classe « GAOptimizer » et la classe « CIndividu »

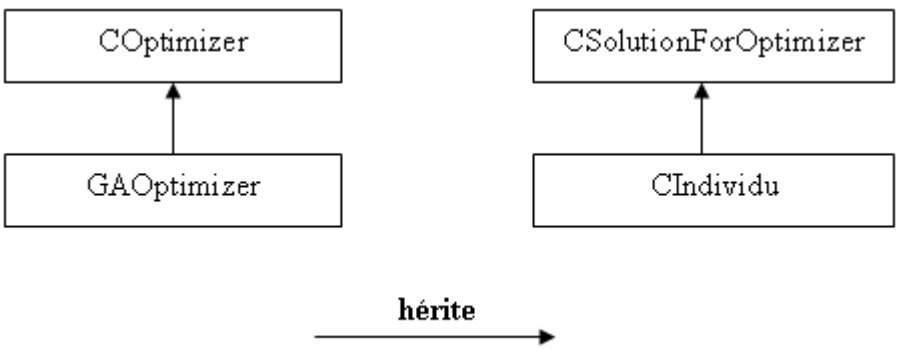


Figure 47. Graphe d'héritage, un individu est une solution et un algorithme génétique est un optimisateur.

La classe « CIndividu »

Un ensemble de comportements (qui sont des opérateurs de mutation) ont été ajoutés, elles ont tous comme paramètre un pointeur vers un GAOptimizer, les mutations opèrent les trois niveaux proposés. Pour chaque niveau il y a plusieurs choix, c'est-à-dire, plusieurs méthodes de mutations implémentées.

La classe « GAOptimizer »

Elle contient les informations suivantes

- La probabilité de mutation et celle de croisement.
- Un pointeur vers une méthode d'évaluation (évaluation avec ou sans trie).
- Un pointeur vers une méthode de sélection.
- Un pointeur vers une méthode de croisement.
- Un pointeur vers une méthode de mutation.
- La population et le bassin génétique (*Voir ci-dessous*).
- Un vecteur contenant les variances des paramètres d'une transformation. Ce vecteur servira pour évaluer l'homogénéité d'une population.

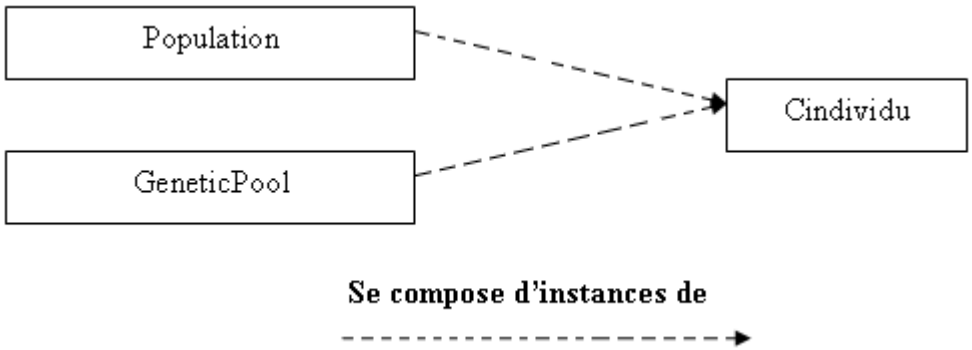


Figure 48. Une population et un bassin génétique sont des ensembles d'individus.

La classe performe les méthodes suivantes

- Pour l'évaluation elles mettront à jours le vecteur des variances,
  - **EvaluationWithSorting**() évalue les individus de la population ensuite les trier.
  - **EvaluationWithoutSorting**() évalue seulement les individus de la population, ils gardent l'ordre initial.
- Pour la sélection les individus sélectionnés seront stockés dans le bassin, elles sont sélection élitiste **ElitistSelection**, par roue de fortune **LottryWheelSelection**, par rang de classement **RankSelection**, par tournois binaires **BinaryTournmentSelection**.
- Le croisement Croiser les meilleurs avec les meilleurs **BestBestCrossed**, les meilleurs avec les mauvais **BestWorstCrossed** ou encore par tournois binaires **BinaryTournmentCrossOver**.

VII.5.1.2. La classe « BGAOptimizer » et la classe « QGAOptimizer »

Une méthode d'optimisation par algorithmes génétiques en codage binaire, et une autre en codage quantique, la première travaille sur la chaîne binaire de l'individu, la deuxième sur sa chaîne quantique. Il n'y a que les fonctions de mutation et de croisement qui changent. Avec l'ajout d'une variable qui exprime le niveau de l'opérateur de croisement (le nombre de points de coupures à générer).

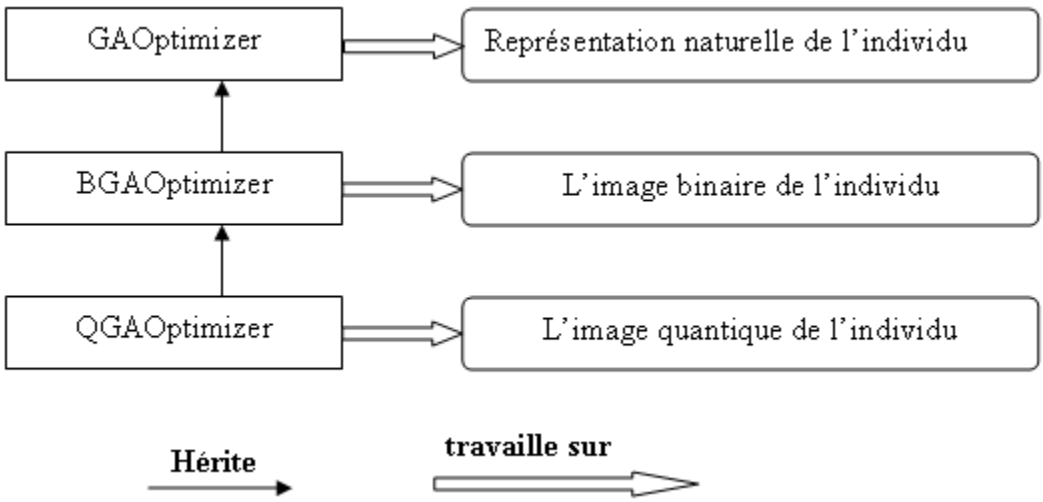


Figure 49.Héritage entre les types d'algorithmes génétiques.

VII.5.2. Mise en œuvre des méthodes « recuit simulé »

VII.5.2.1. La classe « RS » et la classe « CEtat »

Comme déjà vu pour les algorithmes génétiques, le recuit simulé a été mis en ouvre (RS), il hérite son comportement directement de la classe COptimizer, pour qu'il puisse s'intégrer facilement dans la bibliothèque des métaheuristiques implémentées, de plus CEtat qui est une solution pour recuit hérite tout de mémé son comportement de CSolutionForOptimize.

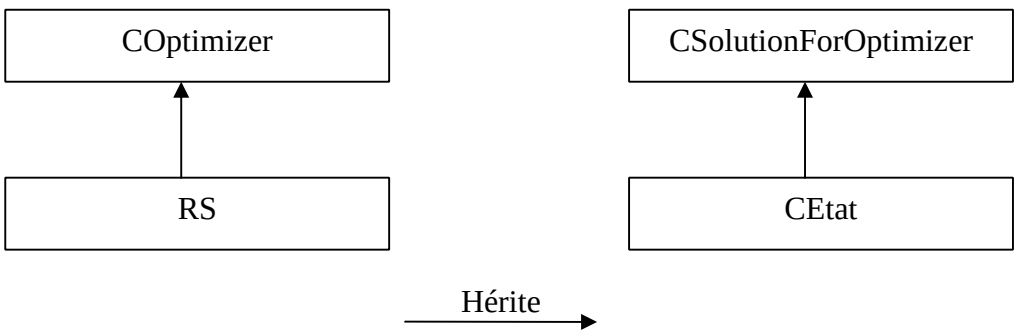


Figure 50. Graphe d'héritage un état est une solution, recuit simulé est une méthode d'optimisation.

La classe « CEtat»

Elle rajoute

- un pointeur vers la méthode membre chargée de la génération d'un état voisin.
- Un pointeur vers l'optimisateur recuit.

Elle définit un ensemble de routines pour la génération des états voisins

- **Gen_ByMutuation()** elle consiste a spécifier aléatoirement des éléments dans le vecteur d'état courant, et changer ces derniers par des valeurs générées aléatoirement dans l'espace de recherche de l'optimisateur pointé.
- **Gen_ByAdd_Random_Vector() &Gen_ByAdd_Normal_Vector()** ajout d'un vecteur suivant la loi uniforne ou normale dont les caractéristiques se trouvent parmi les propriétés de l'optimisateur pointé.
- **Gen_ByRandomVector() & Gen_ByNormalVector()** elle consiste à sélectionner un élément, et de lui affecter une valeur aléatoire dans l'espace de recherche. Les éléments seront sélectionnés au tour de rôle
- **Gen_Independently()** génération d'une nouvelle solution.

La classe « RS »

Elle contient les informations suivantes

- Les différents type d'états Etat_Courant,Etat_Opt,Etat_Voisin.
- La température initiale T_Init, température courante T_Courante, Alpha la constante de refroidissement de T_Courant.
- Un pointeur vers une méthode de refroidissement de T_Courant.
- Un pointeur vers une méthode d'initialisation de la température.
- Un pointeur vers une méthode de fin de recherche.

VII.5.2.2. La classe «Binary_RS »

Une méthode d'optimisation par recuit simulé en codage binaire, s'avère très intéressante dans le cas d'étude du comportement de son processus d'optimisation.

Le codage binaire na d'influence sur recuit simulé que au niveau d'exploration de l'espace de recherche, le schéma classique de la méthode ne change pas, c'est que la classe CEtat qui change, au niveau de ces routines qui génèrent des nouveaux états voisins, car le nouveau codage influe sur la manier de modifier un ancien état.

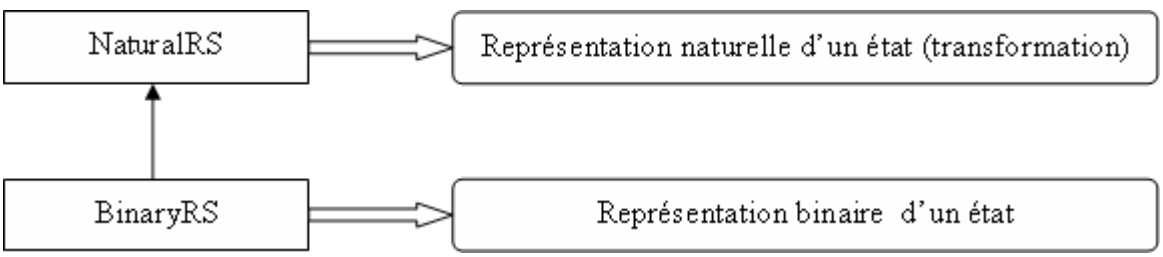


Figure 51. Recuit simulé binaire et naturel, et les types des états qu'ils manipulent.

La classe « CEtat_binary »

Elle définit une fonction generate() pour la génération d'états voisins. Generate () consiste à faire spécifier aléatoirement des bits dans la chaîne binaire de CEtat_binary, et inverser ces bits.

Réorganisation des objets

Notre solution au problème (Recalage des Images satellitaires, une approche basé sur métaheuristiques (RS & AG) est entièrement basée sur une modélisation orienté objet ce qui fait que les entités du problème sont découpé en classe ou objet (modèle dynamique) selon la nature ou leur rôle dans le programme, entre autre la conception envisagée rend notre programme plus flexible et elle lui permet une mise à grappe de nouveaux modules aisée pour être exploite par la suite pour autre problème.

Par initiative la définition du problème que nous voulons résoudre, manipule des images, c'est à dire qu'il faut concevoir comme première classe la classe IMAGE,

De plus notre solution opte a résoudre notre problème par le biais des optimisateurs (tel que RS et AG) donc une mise en place d'une classe OPTIMISATEUR s'impose, la classe SOLUTION qui représente comme son indique une solution pour le problème en question, par ailleurs comme tout programme, une panoplie de classes va être crée au fil de l'eau de la mise en œuvre du programme, parmi lesquelles sont les classes de gestion d'entrée sortie ou d'interface, les classes de sauvegarde de paramètres et enfin les classes qui serrent a la définition du problème lui même.

Ci-après nous allons énumérer les différentes classes du problème.

Classe CIMAGE

La classe CImage modélise le minimum d'outils ou de fonctions qui manipulent une image pris dans son format conventionnel propre à notre problème.

Ci-après le résume des fonctions et les membres de la classe :

- **Membres**
  Nombres de colonnes et de lignes pour définir la résolution de l'image.
  Image_couleur pour avoir l'état d'affichage de l'image (Gris ou couleur).
  Bornes de la zone d'intérêt (x min et max, y min et max).
  Les points de références (point A, B et C).
  Pixels un vecteur qui contient l'image dense et
  pixels_couleur pour l'image couleurs

- **Fonctions**
  Les fonctions d'interrogations des vecteurs de l'image (les Gets et les Sets).
  Les fonctions de transformations de l'image.
  Les fonctions de calcul de ressemblance.
  Les fonctions de lecture et d'affichage des images.

Classe CSolution

La classe CSolution représente la solution apportée au problème, dans notre cas la solution c'est la transformation de l'image recherche pour le problème de recalage, rappelons que la modélisation de notre solution est en orienté objet ce qui fait que la classe CSolution est la squelette ou le modèle mère pour chaque type de transformation de l'image ainsi qu'une simulation de la résultante d'un optimisateur (énergie, individu…).

- **Membres**
  Vecteur de solution.
  Taille du vecteur.
  Nature de solution.
  Type de transformation
- **Fonction**
  Initialisation de la solution selon le type de transformation.
  Opération d'addition, de soustraction,...
  Calcule de qualité de recalage.

Classe COptimisateur

La classe COPTIMISATEUR c'est le modèle abstrait des optimisateurs implémentés dans notre solution, il contient des membres et des fonctions de base pour une mise en œuvre aisée des métas heuristique, ce qui fait que chaque nouvel optimisateur implémenté dans notre cas hérite obligatoirement de cette classe pour qu'il puisse être joignait a notre programme.

- **Membre**
  Configurations de l'optimisateur
- **Fonction**
  La méthode "**optimiser"** elle a pour but de lancer la procédure de recherche et d'optimisation.

Classe CRecalage

La classe CRECALAGE elle embarque en elle, tous les paramètres liés au problème de recalage.

- **Membre**
  Type de recalage.
  Méthode d'optimisation.
  Configuration de l'optimisateur.
  La solution optimale.
  L'espace de recherche.
- **Fonction**
  Initialisation de tous les paramètres avec des valeurs par défauts.

Classe GenNBAleat

La classe GenNBAleat génère une liste de nombre aléatoire selon le choix de loi de probabilité ainsi les bornes de l'espace de recherche.

- **Membre**
  Liste de génération des nombres aléatoires
  La loi de probabilité.
  L'espace de génération des nombres aléatoires.
- **Fonction**
  Initialisation de la classe.
  Génération des nombres.

Classe RS

Recuit simulé c'est un optimisateur ou une méta heuristique, qui a été implémentée dans notre travail pour résoudre le problème de recalage. Elle hérite directement de la classe COPTIMISATEUR, elle a comme particularité des paramètres en plus, les membres liés a la simulation cette dernières, ce qui fait que sa structure interne change vis à vis des autres classes qui héritent de la classe mère COPTIMISATEUR.

- **Membre**
  Température initiale et courante.
  Solution courante, et voisine.
  Nombre d'étapes et d'itération par étapes.
- **Fonction**
  Initialisation de la température initiale.
  Indicateur de fin de recherche.
  Critère de Metropolis.
  Génération de nouvelles solutions.
  Décroître la température courante.

Classe AGs

La classe AGS pareil que la classe RS hérite directement de la classe COPTIMISATEUR.

Cette classe contient comme paramètres en plus de la classe mère les paramètres conduits à la simulation de la classe.

- **Membre**
  Taille de la population.
  Nombre de génération max.
  Taux de croisement et de mutation.
- **Fonction**
  Initialisation du bassin génétique.
  Méthode de croisement.
  Méthode de sélection.
  Méthode de mutation.
  Indicateur de fin de recherche.

Classe CConfigParallele

La classe CCONFIGPARALLE hérite directement de la classe CRecalage, elle ajoute comme nouveaux membres le délai de synchronisation de processus ainsi que les plages de paramètres pour les liaisons directe avec les autres processus concurrents.

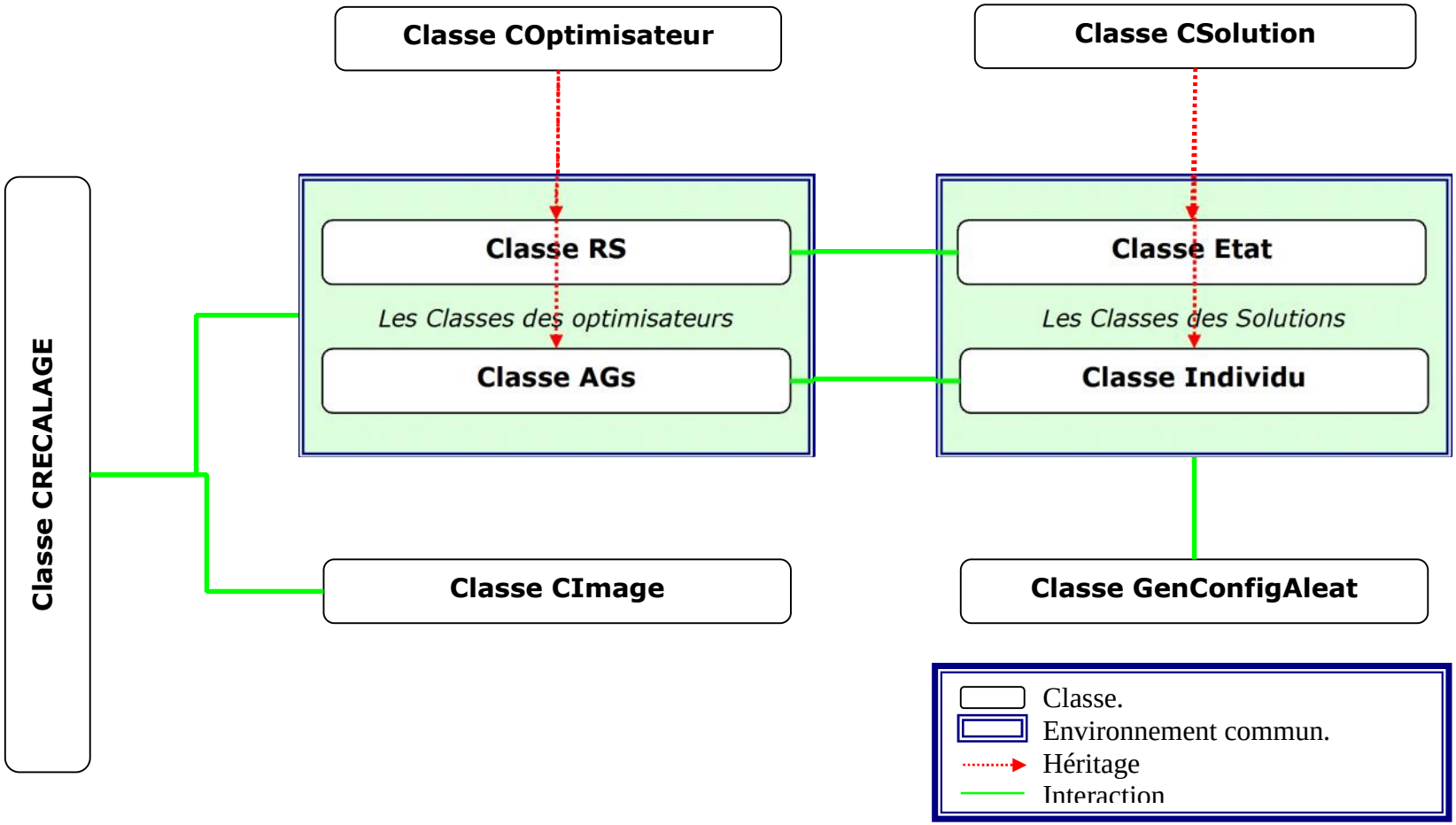
Cette classe est purement abstraite ce qui fait que pour l'implémentée il faut prendre en considération les protocoles réseaux moyennés ainsi que l'architecteur ou la topologie de la solution réseau (architecture maître esclave, ou point à point ….).

Classe COptimisateurParallele

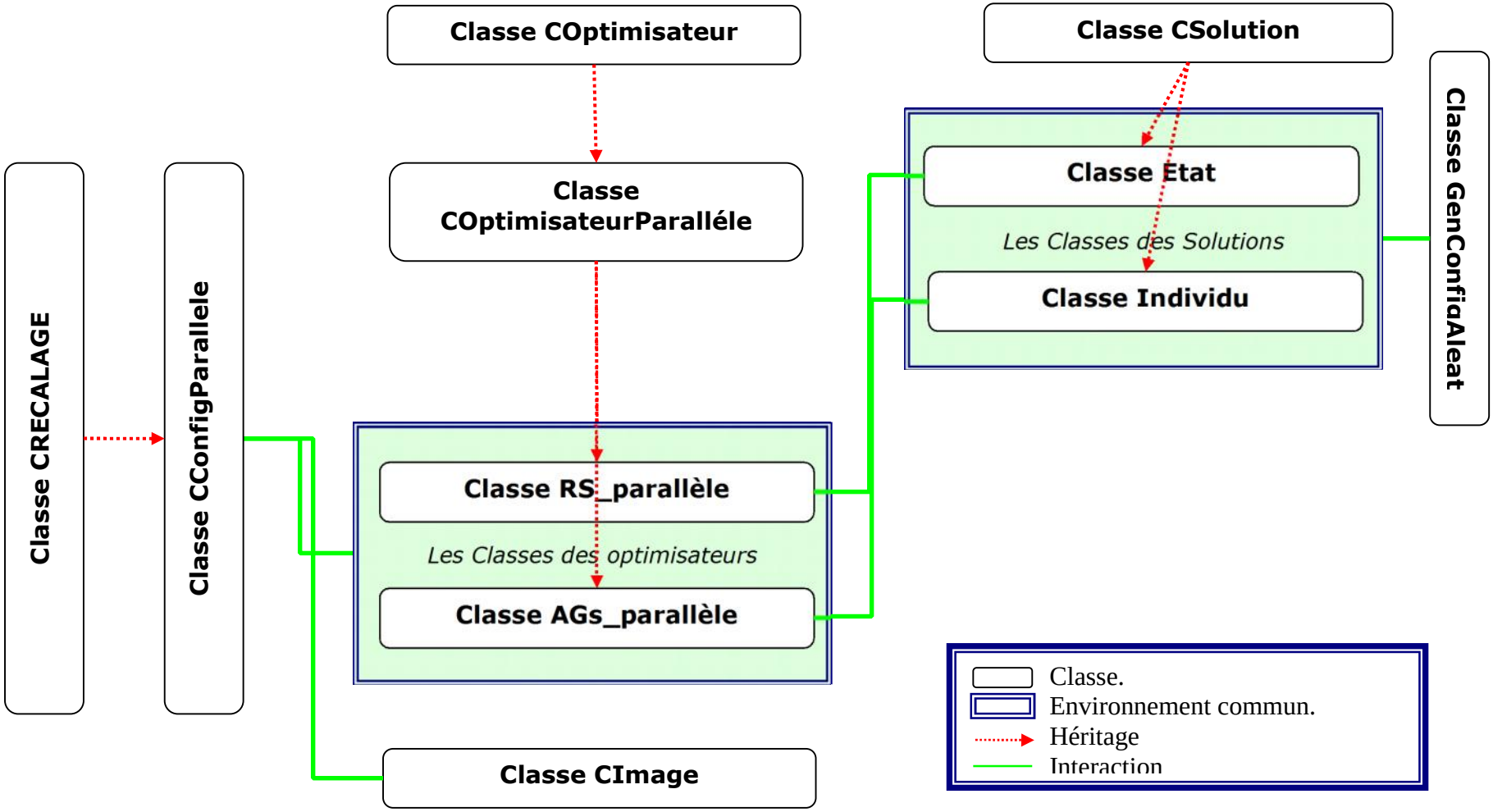
La classe COptimisateurParallele hérite de la classe COPTIMISATEUR, a l'exception que cette classe elle prend en charge la gestion de synchronisation entre processus ce qui fait qu'elle suspend la recherche a tout moment, et elle reprend qu'on la synchronisation est faite.

Cette classe elle aussi est abstraite, cela dit qu'il y'a une dépendance étroite entre l'implémentation et les limites matériels ou plus généralement l'environnement d'implémentation.

VII.6. SCHEMA CONCEPTUEL REDUIT DU NOYAU EN MODE SEQUENTIEL



VII.7. SCHEMA CONCEPTUEL REDUIT DU NOYAU EN MODE PARALLELE



VII.8. CONCLUSION

A travers ce chapitre nous avons, d'abord, validé théoriquement l'adéquation des métaheuristiques au problème de recalage d'images. Deux métaheuristiques ont été proposées pour le résoudre, avec les possibilités de codages courantes.

Dans le but de valider expérimentalement et définitivement cette adéquation, nous avons développé une plateforme puissante «RIS-MH». Elle prend en charge 03 types de codage (naturel, binaire et quantique) pour représenter une solution, et qui permet l'intégration intuitive de n'importe autre méthode d'optimisation tout cela grâce à la séparation sémantique des entités et la conception orientée objet des différentes entités de la plateforme. Encore et grâce à la méthode d'évaluation adoptée, les bibliothèques des méthodes d'optimisation peuvent être exploitées afin de résoudre n'importe quel problème d'optimisation dont on manipule des solutions avec leurs représentations naturelles, tous les problèmes dont les solutions binaires (ou quantiques) peuvent être codées sur moins de 64bits.

Il ne nous reste, à présent, que la validation expérimentale du modèle de recalage à base de métaheuristiques, qui sera le but du chapitre suivant.

CHAPITRE VIII. TESTS ET RESULTATS

VIII.1. INTRODUCTION

Tous les éléments ont été collectés pour analyser expérimentalement les résultats qui vont être donnés par le modèle de recalage proposé. En effet, durant cette partie, on va tester les méthodes d'optimisation étudiées, analyser et commenter les résultats apportés par chaque méthode d'optimisation.

En ce qui concerne l'environnement d'exécution, tous les tests ont été effectués sur un ordinateur personnel dont les caractéristiques sont

1. Matériel processeur Intel Pentium 4 (1GHz), 256Mo du RAM.

2. Système d'exploitation Microsoft Windows XP service pack2.

Etant donné que le coût de calcul fut parmi les paramètres critiques qui valident la performance d'une solution informatique implémentée pour résoudre un problème d'optimisation, et pour analyser correctement la relation entre les différentes entités et le coût relatif à chacune, nous n'avons considéré que le temps consommé par la méthode d'optimisation et ses entités. Les résultats de ce chapitre ne comptabilisent pas le coût des traitements des entrées/sorties.

VIII.2. DEFINITIONS

- Les paramètres génétiques (ou une configuration génétique) sont les paramètres d'un algorithme génétique le critère de fin de recherche, l'opérateur de sélection, de croisement, le taux de mutation, et évaluation ainsi le critère de similarité…etc.

- Les paramètres recuit (ou une configuration recuit) c'est l'ensemble des paramètres d'une méthode d'optimisation recuit simulé.

- Une configuration (génétique ou recuit) perdante c'est une configuration qu'on a peu de chances de converger vers des bonnes solutions, contrairement avec la configuration gagnante, on a la chance de converger vers de très bonnes solutions

- Une configuration intermédiaire, il y a de grandes chances de converger vers des résultats moyens (pas bons et pas mauvais), elle est comme une position stratégique, à partir de laquelle on peut constater les améliorations et les diminutions de performances du recalage suite à une modification des paramètres.

VIII.3. METHODOLOGIE DES TESTS

Pour être cohérents et efficaces, nous avons suivi les étapes suivantes pour valider un modèle de recalage à base d'une métaheuristique

- Choix des paramètres optimaux pour chaque méthode d'optimisation afin d'assurer une configuration gagnante.
- Adopter ces paramètres pour résoudre les problèmes de recalage réels.

VIII.3.1. Choix des paramètres optimaux

Les deux méthodes étudiées et proposées sont des méthodes paramétriques, dont le choix des paramètres ne se fait que par « tâtonnement » (i.e. faire beaucoup de tests pour estimer une valeur adéquate pour chaque paramètre), notre stratégie consiste à faire un maximum de tests sur des problèmes dont on connait le meilleur résultat.

Cette étape va se dérouler en deux parties : la recherche des paramètres optimaux, ensuite la validation du choix.

VIII.3.1.1. recherche des paramètres optimaux

On va essayer d'étudier la variation de performances (coût de calcul et qualité) par rapport à la variation des paramètres de la méthode d'optimisation considérée. Il faut dire qu'à ce stade, les méthodes d'optimisation adoptées suivent des schémas stochastiques (i.e. il y aura forte chance de converger vers des résultats différents après deux exécutions consécutives), donc il est judicieux d'estimer une probabilité de convergence vers une certaine classe de résultats, cette approche nous permettra un meilleur contrôle des tests et une meilleure interprétation des résultats.

Cette étape consiste en un ensemble de tests qui ont pour but la recherche des paramètres optimaux de chaque méthode, ces tests vont se dérouler (pour résoudre le même problème de recalage avec les mêmes images et la même transformation à chercher) la façon suivante :

- Prendre une configuration intermédiaire ;
- Varier un seul paramètre à la fois, pour chaque valeur lancer 20 fois le processus de recalage ;
- Etudier les résultats.
    - **Evaluer la qualité** de recalage.
    - **Estimer la probabilité** de convergence vers une certaine classe de résultats.
    - **analyser le coût** en temps de calcul machine.

Tous les tests ont été effectués pour résoudre un problème de recalage dont l'image à recaler n'est autre que la transformée de l'image consigne. La transformation de paramètres (-12, 10, 0.2) est rigide. Donc la meilleure transformation à chercher est la transformation inverse (12,-10, -0.2). (Voir ci-dessous)

Image consigne « résolution (1024*768) » Image à recaler « résolution (1024*768) »

Figure 52. Image 1 : problème à résoudre, pour le choix des bonnes configurations génétiques.

Evaluer la qualité de recalage

Pour évaluer la qualité de la meilleure solution délivrée par un processus d'optimisation, généralement, on utilise des formules exprimant la distance entre deux solutions. Ces formules sont applicables si toutes les variables d'une solution varient dans des espaces de même échelle, ce n'est pas le cas avec les variables d'une solution d'un problème de recalage. Par exemple une transformation rigide comporte 3 variables (T1 et T2 (en pixels): translations, Alpha (en radians) angle de rotation), les deux premières variables ont la même nature et exprimées avec la même échelle, contrairement à la troisième qui a une nature intégralement différente, et dont l'échelle est trop petite.

Pour cela, nous avons adopté un autre critère pour évaluer la qualité de recalage à travers la meilleure transformation délivrée. En effet, on va utiliser un critère de similarité (le rapport de corrélation) entre l'image recalée (avec la meilleure transformation fournie par la méthode) et l'image consigne.

Pourquoi le rapport de corrélation ?

Comme nous l'avons déjà présenté, on a implémenté plusieurs critères de similarité l'entropie, l'erreur quadratique, la covariance et le rapport de corrélation. Avec les trois premiers nous n'avons aucun moyen d'estimer l'écart entre les deux images d'une façon interprétable, en effet les trois méthodes fournissent des mesures dont la limite idéale n'est pas connue (la covariance), et si cette limite est connue on n'arrive pas à interpréter les résultats, c'est le cas de l'entropie et l'erreur quadratique qui tendent vers zéro dans le cas de similarité absolue. Contrairement au rapport de corrélation, qui est une mesure standardisée, bien limitée (entre 0 et 1) et on arrive facilement à l'interpréter.

Il faut dire que nos observations des résultats de recalages, nous ont guidé vers une classification des recalages selon la qualité, la classification est basée sur le calcul du rapport de corrélation entre l'image consigne et l'image résultat de recalage. Donc le rapport de corrélation est comme suit :

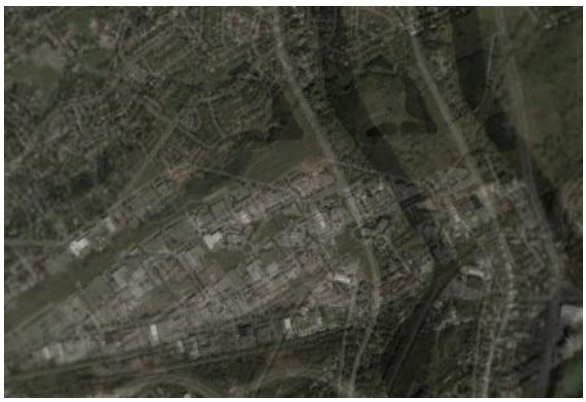
- **Inférieur à 0.5 (inacceptable):** le recalage est inacceptable, tous les paramètres de la meilleure transformation délivrée par le processus de recalage sont loin de celles de la transformation optimale.

- **Entre 0,5 et 0,75** on remarque que le processus d'optimisation arrive à trouver une variable optimale, généralement l'angle de rotation, les autres sont relativement loin des paramètres optimaux.

- **Entre 0.75 et 0.85 (qualité moyenne)** la distance entre la meilleure transformation et l'optimal est relativement petite. Le recalage est moyen, car il n'arrive pas à aligner correctement les contours.

- **Entre 0.85 et 0.95 (bonne qualité):** encore plus petite, la distance entre les deux transformations. La majorité des résultats sont pratiquement acceptables.

- **Supérieur à 0.95 (qualité excellente):** le cas idéal en une recherche stochastique dont les résultats sont excellents.



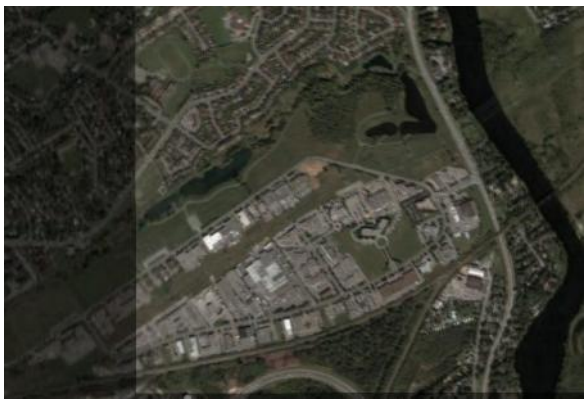
1. **Rapport de corrélation=0.39 : aucun paramètre n'est aligné.**

2. **Rapport de corrélation=0,79 (tous les paramètres sont proches des optimaux)**

3. **Rapport de corrélation=0,55 : un seul paramètre est aligné (translation en axe des X).**

4. **Rapport de corrélation=0,67 : (seule l'angle de rotation est alignée).**

5. **Rapport de corrélation=0,87 : fusion acceptable.**



6. **Rapport de corrélation=0,98. fusion parfaite.**

Figure 53. Le rapport de corrélation un bon indicateur de la qualité de recalage.

VIII.4. VALIDATION DU CHOIX DES PARAMETRES

Le choix des paramètres, durant la partie précédente, se faisait sur la base d'un seul problème de recalage défini avec les mêmes images et la même transformation à chercher. Il faut se dire si ce choix est judicieux pour tous les problèmes de recalages ? Cette étape a pour but, la validation du choix à travers une multitude de tests sur d'autres problèmes avec diverses images et diverses transformations à chercher. Les solutions optimales de ces problèmes sont toutes connues.

VIII.5. EXECUTION DES ALGORITHMES GENETIQUES

VIII.5.1. Réglage des paramètres génétiques

On rappelle que cette étape consiste à identifier une configuration génétique gagnante, avec laquelle on a grande chance de converger vers des bons résultats en termes de qualité de recalage (rapport de corrélation> 085).

Pour cela, et pour estimer la probabilité de convergence vers une classe de résultats, chaque configuration sera testée 20 fois. Ça nous permettra d'estimer cette probabilité à un ordre d'erreur de 1/20 = 0.05, qui est suffisamment petit. On rappelle aussi que chaque configuration testée sera à base d'une configuration génétique intermédiaire, cela nous facilitera le contrôle et l'interprétation des résultats.

Premièrement on va essayer d'observer (plus précisément démontrer) l'impact de la taille du domaine de recherche sur la qualité de recalage, de remédier à ce problème par l'augmentation de la taille de population et d'estimer le surcoût dans une seconde étape, et pour finaliser faire quelques tests pour choisir les bonnes opérateurs génétiques (la sélection et la croisement).

VIII.5.2. Taille de l'espace de recherche et qualité de la recherche

Clair est le fait qu'un algorithme génétique suit un schéma d'optimisation descendant, i.e. la chance de trouver la bonne solution vient en collectant les gènes optimaux des solutions moins bonnes moyennant l'opérateur de croisement, autrement et si ce dernier sera incapable de combiner les gènes optimaux alors le processus n'a aucune chance de trouver la solution optimale. Cela est dû à deux facteurs, l'un des deux est la taille de l'espace de variation des solutions.

Dans notre premier test on va essayer de démontrer se qu'on vient de proposer, et d'en tirer des conclusions

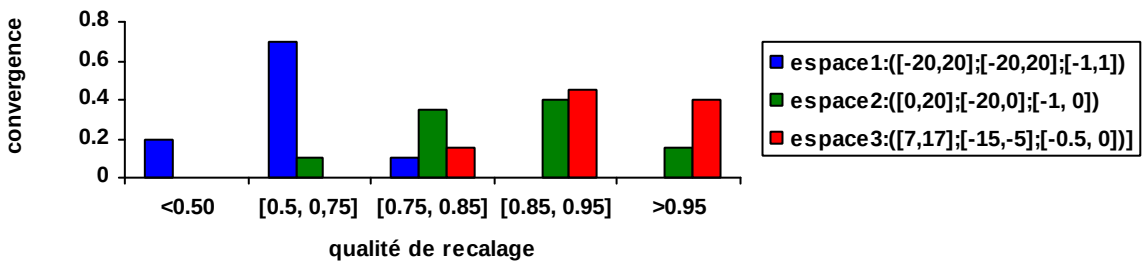


Figure 54. Graphe 3: qualité de recalage vis-à-vis la taille de l'espace de recherche.

La configuration stable de l'optimisateur génétique est la suivante

Taille de population =10. Sélection déterministe. Croisement meilleur/meilleur. Probabilité de croisement=0.9 Probabilité de mutation=0.1 Critère de similarité rapport de corrélation.

On remarque que dans un espace de recherche étendu la probabilité de converger vers une bonne solution est typiquement nulle (théoriquement trop petite). Cela est le cas du premier espace (couleur bleue) dans le quelle la qualité de recalage dans les plus part des cas mauvaise (inférieur à 0.75). Plus on restreint l'espace de recherche plus on augmente la probabilité de converger vers une bonne transformation, cela est le cas des recalages sur les deux autres espaces (vert et rouge). Cela était prévu !

VIII.5.3. Taille des populations, qualité de recalage et le surcoût

Il n'est toujours pas facile de pouvoir limiter l'espace de recherche, car généralement on cherche une méthode de recalage automatique pour ne pas s'amuser à initialiser l'optimisateur pour chaque couple d'images. L'alternative (pour assurer la convergence vers une bonne solution) est d'augmenter la taille des populations, qui est le deuxième facteur contrôlant la collection des gènes optimaux.

Pour bien toucher les résultats on propose d'améliorer les résultats de recalage sur l'espace1 du test précédent

Espace [-20, 20] ; [-20, 20] ; [-1, 1]. Sélection déterministe. Croisement meilleur/meilleur. Probabilité de croisement=0.9 Probabilité de mutation=0.1 Critère de similarité rapport de corrélation.

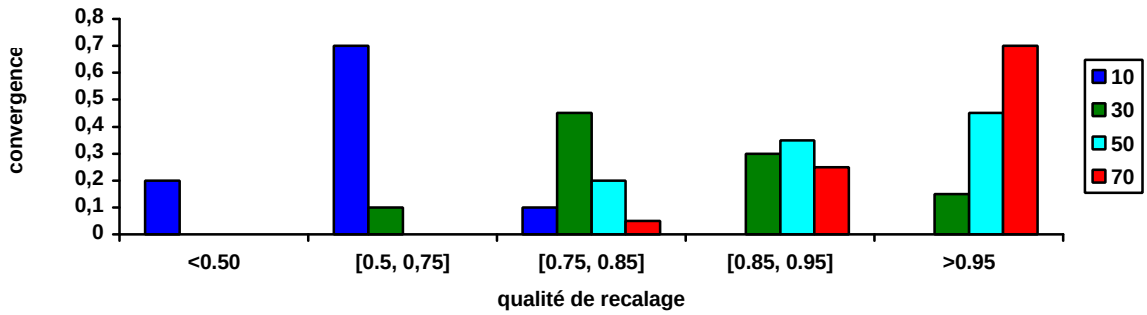


Figure 55. Graphe 4 : qualité de recalage vis-à-vis la variation de la taille des populations.

Evidement, et ce qui était prévu, plus on augmente la taille de la population plus on augmente la probabilité de convergence vers des résultats meilleurs. Mais cette fois-ci il ne s'agit pas seulement de démontrer cette proposition qui est évidente, mais il faut bien étudier le comportement du recalage vis-à-vis la variation de ce paramètre.

Pour un recalage de bonne qualité, la meilleure solution doit présentée une qualité supérieure à 0.85. Sur le graphe on remarque qu'avec plus de 50 individu par population, la probabilité de converger vers une très bonne solution est suffisamment grande, cette probabilité tend vers l'idéale « 1 » avec 70 individu par population, et avec cette valeur que la majorité des recalages présentaient des qualités supérieures (rapport de corrélation >0.95).

Evidement, il y a un autre coût proportionnel à la variation de la taille des populations. On remarque que le coût de calcul s'accroît rapidement avec la croissance du paramètre qu'on vient de tester (voir Graphe 5). Cette croissance est causée par deux facteurs le traitement de plus d'information (i.e. l'évaluation de plus d'individus) et le tri des populations.

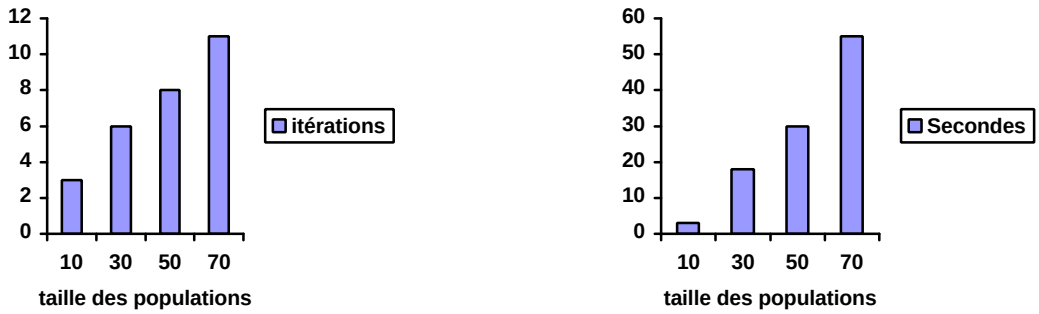


Figure 56. Graphe 5 : Temps de réponse en fonction de la taille de la population.

VIII.5.3.1. L'impact des opérateurs de sélection

Comme nous l'avons déjà mentionné, nous avons implémenté quatre critères de sélection

- **Sélection déterministe** qui choisit la meilleure moitié de la population d'une façon élitiste.
- **Sélection par roue de loterie** un schéma de sélection stochastique, dont la probabilité de sélection dépend d'une façon directe de la fitness des individus.
- **Sélection par rang de classement** encore un schéma de sélection aléatoire, mais dans ce cas précis, la probabilité de sélection dépend du rang de l'individu dans sa population (dépend d'une façon indirecte à sa fitness).
- **Sélection par tournoi binaire** deux individus sont tirés aléatoirement, on retient le meilleur et on rejet le mauvais.

On préfère étudier la sélection par tournoi binaire isolément, en concentrant toute notre intention sur les trois premières. Tout d'abord on rappelle que la sélection par roue de la fortune et par rang de classement, ont été implémentées pour remédier au problème de convergence prématurée causé par la sélection déterministe. Dans ce paragraphe on touchera cette cause

Pour les tests, et concernant notre fameuse « position intermédiaire » on propose la configuration génétique suivante

Espace [-20, 20] ; [-20, 20] ; [-1, 1]. 30 individus par population. Croisement meilleur/meilleur. Probabilité de croisement=0.9 Probabilité de mutation=0.1 Critère de similarité rapport de corrélation.

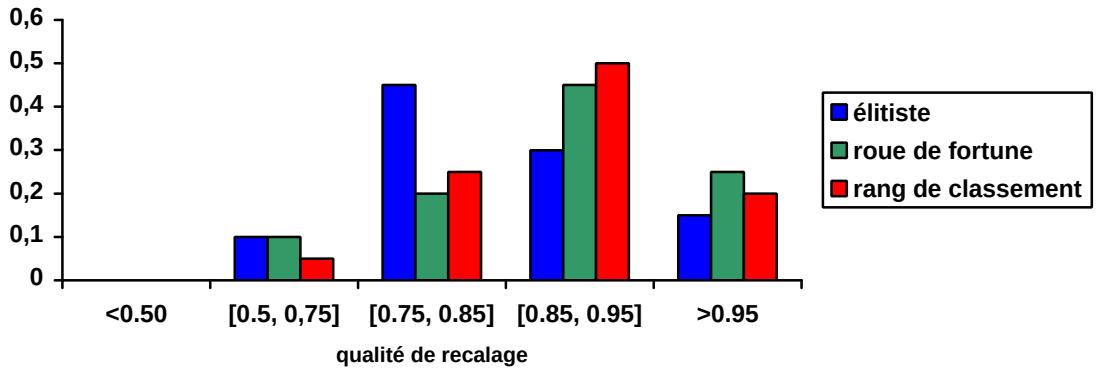


Figure 57. Graphe 6 l'impact des opérateurs de sélection sur la qualité de convergence.

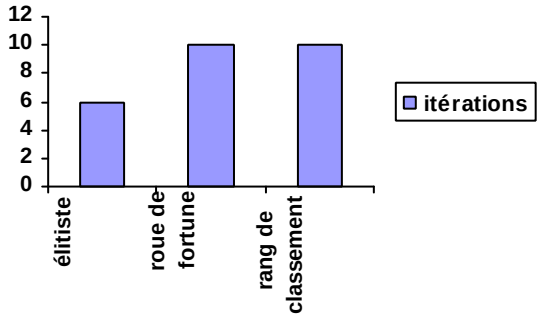


Figure 58. Graphe 7 Coût des calculs en fonction des sélections.

Sur le (Graphe 6) on observe une amélioration remarquable (en termes de qualité de convergence) apportée par les sélections aléatoires (par roue de fortune et par rang de classement) relativement aux résultats fournis par un processus de recalage basé sur sélection élitiste. Cette amélioration est due au retard de convergence causé par une sélection probabiliste, le (Graphe 7) témoigne pour cette proposition. On remarque aussi que la sélection par roue de fortune et celle par rang de classement présentent pratiquement les mêmes performances en terme de qualité de convergence, si on ne tient pas compte de l'amélioration (qu'on pense qu'elle pratiquement négligeable) de la sélection par rang de classement. Cette amélioration nous conduira vers la proposition que la sélection purement aléatoire peut encore améliorer la performance du recalage, c'est pour cette idée là que nous avons préféré de retarder l'étude de la sélection par tournoi binaire.

En termes de coût de calcul, il est évident qu'il va augmenter avec le retard de convergence. Cependant, le surcoût n'est pas dû au traitement des itérations supplémentaires seulement, mais aussi au calcul des probabilités de sélections.

VIII.5.4. L'impact des opérateurs de croisement

Notre bibliothèque des algorithmes génétiques fournit plusieurs possibilités de croisement entre individus, le croisement du meilleur individu avec son successeur, celui du meilleur avec le mauvais ou encore le croisement par tournois binaires. Naturellement, les deux premiers opérateurs nécessitent un tri des individus dans le bassin génétique, le troisième ne nécessite pas ce traitement supplémentaire. Il faut dire que les opérateurs de sélections maintiennent l'ordre des individus s'ils nécessitent le tri (sélection élitiste, par roue de fortune, par rang de classement) la sélection par tournois binaires ne garantit pas l'ordre des individus dans le bassin. Donc on peut dire que le croisement meilleur/meilleur et meilleur/mauvais marchent avec la sélection déterministe, par roue de fortune ou par rang de classement. L'étude du croisement par tournois binaires se fera avec l'étude de la sélection par tournois binaires.

Le but de ce paragraphe est d'analyser l'impact de cet opérateur sur, toujours, la qualité de recalage et le temps de calcul. En choisissant la configuration, qui a démontrée une qualité intermédiaire, suivante

Espace	[-20, 20] ; [-20, 20] ; [-1, 1]
Individus par population	30
Sélection	élitiste
Probabilité de croisement	0.9
Probabilité de mutation	0.1
Critère de similarité	rapport de corrélation

Tableau 8. Paramètres tests opérateur de croisement.

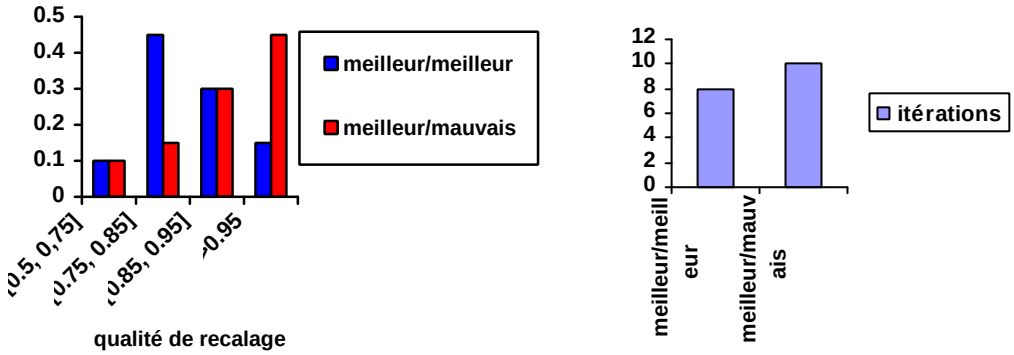


Figure 59. Graphe 8 : croisement meilleur/mauvais et convergence.

Les mêmes remarques que les précédentes s'appliquent sur cet opérateur concernant la qualité de recalage et le temps de calcul, en effet le croisement meilleur/mauvais provoque un retard de convergence remarquable, cela est dû au partage des gènes non semblables.

Mais il faut dire que dans les plus parts des tests qu'on a fait il parait que le croisement meilleur/meilleur assure la convergence vers des solutions semblables, contrairement au croisement meilleur/mauvais qui assure par fois une bonne convergence mais vers des solutions relativement différentes.

VIII.5.5. Etude des tournois binaires

Durant nos précédents tests nous avons remarqué que le tri des populations entraîne un surcoût non négligeable, si on exempte cette tâche, on aura gagné le temps relatif.

Vu que les tournois binaires sont des schémas stochastiques d'évolution, on aura tellement de peines d'obtenir des résultats de même ordre (pour ne pas dire les même résultats) après deux tentatives d'optimisation, contrairement aux autres opérateurs qui ont prouvé une certaine stabilité de convergence dans le sens où après deux exécutions il y aura grande chance de trouver des résultats très proches.

L'idée est de compenser les limites des opérateurs d'évolution par tournois binaires (qualité de convergence) en exploitant le temps gagné pour le traitement de plus d'individus par population.

Dans ce paragraphe on va surtout essayer d'étudier le comportement des recalages basés sur ces opérateurs vis-à-vis la variation de la taille de la population, qui peut améliorer la qualité de convergence du processus de recalage en utilisant la configuration génétique suivante

Espace	[-20, 20] ; [-20, 20] ; [-1, 1]
Sélection	Tournoi binaire
Croisement	Tournoi binaire
Probabilité de croisement	0.9
Probabilité de mutation	0.1
Critère de similarité	rapport de corrélation

Tableau 9. Paramètres tests tournois binaires.

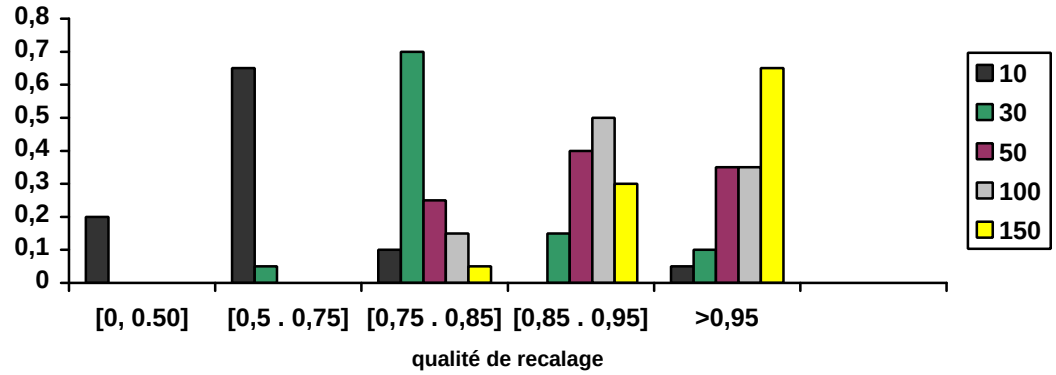


Figure 60. Graphe 9 : qualité de convergence de recalage basé sur une évolution purement aléatoire.

On sait que les opérateurs génétiques par tournois sont des opérateurs d'évolution purement aléatoire, donc on s'attend que les résultats soient tous sauf stables (en terme de qualité), cela peut être démontrée par le (Graphe 9). En effet, avec 10 individus par population il est évident que le résultat de recalage soit mauvais mais parfois on peut converger vers des bonnes solutions (voir l'intervalle >0.95 la couleur noir). Donc il faut être absolument prudent, car comme il y a de la chance de converger vers des excellentes solutions en utilisant une configuration génétique perdante, on peut tout de même converger vers une mauvaise solution en utilisant une configuration gagnante.

On remarque qu'en augmentant le nombre d'individus par génération on augmente la probabilité de converger vers des bonnes solutions, cette probabilité est acceptable dans le cas d'utilisation de 50 individus par population, encore meilleure à 100 individus. Cela est évident.

Le coût en temps de calcul s'accroît avec la croissance du nombre d'individus, cela à cause du traitement d'un nombre important d'individus.

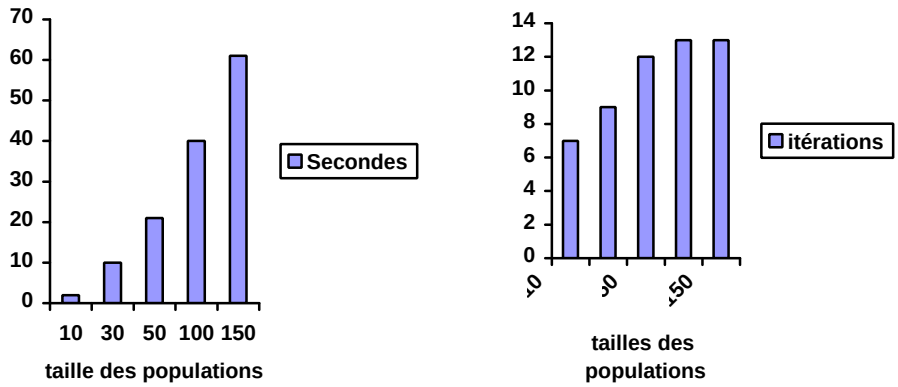


Figure 61. Graphe 10 : coût des calculs en fonction de la taille des populations.

A titre de comparaison, entre l'impact des opérateurs par tournois binaires et les autres

- **En terme de qualité de recalage** on pense que les résultats sont de même ordre de grandeur avec une petite amélioration apportée par opérateurs non purement aléatoires (raffinement par sélection par tournoi binaire et par rang de classement et croisement meilleur/mauvais).
- **En termes de temps de calcul** les résultats sont incomparables, les opérateurs purement aléatoires nous accordent un gain remarquable. Le (*Graphe 11*) est une comparaison selon le coût de calcul entre une configuration purement aléatoire (sélection et croisement par tournois binaires) et la configuration la moins coûteuse en calcul (sélection déterministe pour exempter le coût de calcul des probabilités et croisement meilleur/meilleur pour ne pas retarder la convergence).

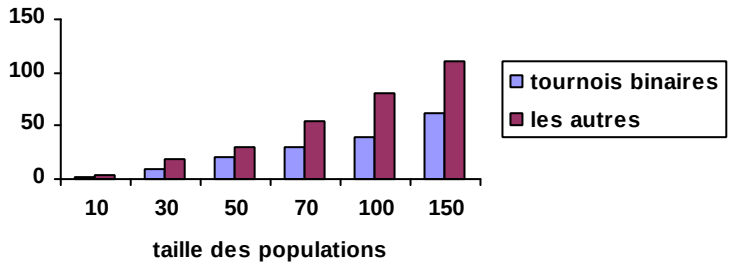


Figure 62. Graphe 11 : l'évolution aléatoire est plus rapide à l'évolution non purement stochastique.

VIII.5.6. Choix du critère de similarité

Nous avons présenté six critères pour mesurer la ressemblance entre deux images, les critères agissant pixel par pixel comme l'entropie, la covariance, l'erreur quadratique et le rapport de corrélation et d'autres par régions l'entropie mutuelle et l'information mutuelle. Le premier genre des critères s'adapte avec excellence au calcul de similarité entre deux images monomodales, l'information mutuelle et l'entropie mutuelles sont des bons indicateurs de ressemblance entre images multimodales.

Dans ce qui suit, on propose d'étudier la qualité de recalage vis-à-vis le choix du critère de similarité, en choisissant la configuration génétique de qualité intermédiaire suivante :

Espace	[-20, 20] ; [-20, 20] ; [-1, 1]
Individus par population	30
Sélection	Elitiste
Croisement	meilleur/meilleur
Probabilité de croisement	0.9
Probabilité de mutation	0.1

Tableau 10. Paramètres tests critère de similarité.

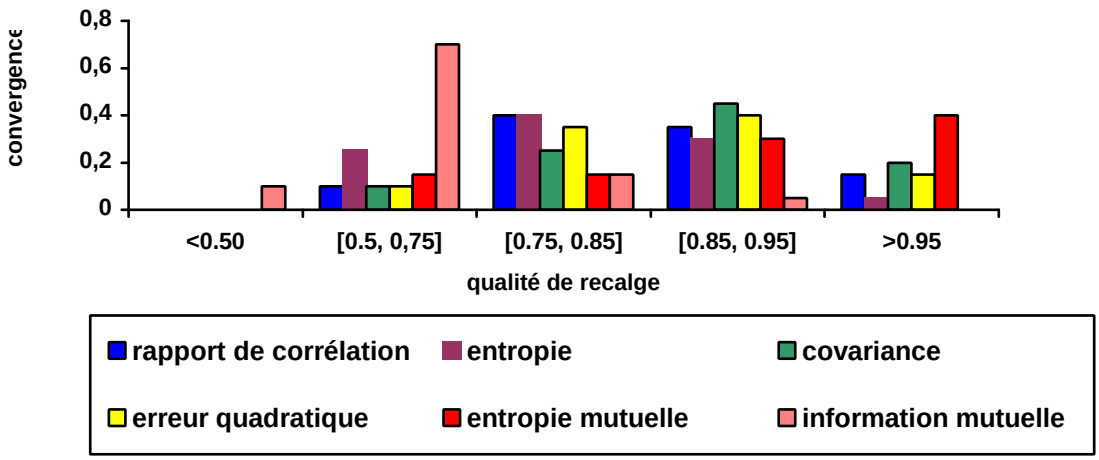


Figure 63. Graphe 12 : l'impact du choix du critère de similarité sur la qualité de recalage.

Les processus de recalage basés sur les critères erreur quadratique, covariance et rapport de corrélation présentent des qualités semblables. En effet, les trois grandeurs reflètent la même information mais dans des échelles distinctes.

On remarque que la qualité de recalage décline en cas d'utilisation de l'information mutuelle, cela est évident car l'information mutuelle délivre des informations plus globales au même temps moins précises. Contrairement l'entropie mutuelle qui accorde les meilleurs résultats.

VIII.5.7. Codage des individus

Trois types de codage on été proposés durant la partie précédente

- Codage naturel.
- Codage binaire.
- Codage quantique.

Malheureusement, l'exécution des algorithmes génétiques en codage binaire et en codage quantiques a décliné, sensiblement, les performances des processus de recalage (en qualité et en coût de calcul) relativement les résultats accordés par le schéma d'évolution génétique naturelle.

On propose d'analyser les résultats de recalages à base des trois schémas algorithmes génétiques (naturels, binaires, quantiques), dont ils disposent de la même taille de population. Nous avons exécuté chaque méthode 20 fois, les graphes (Graphe 13, Graphe 14) vont présenter les meilleurs résultats des algorithmes génétiques binaires et quantiques et les mauvais résultats des algorithmes génétiques naturels.

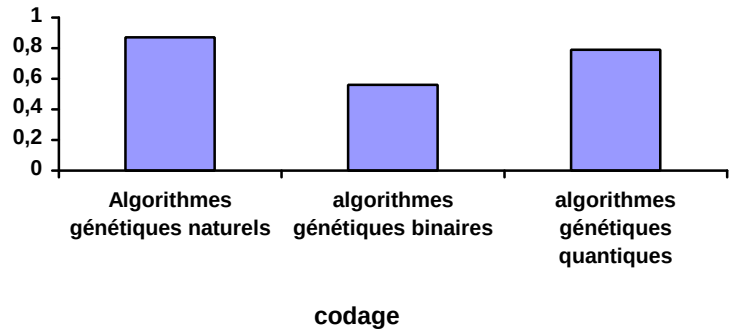


Figure 64. Graphe 13 : qualité de recalage vis-à-vis le codage des individus.

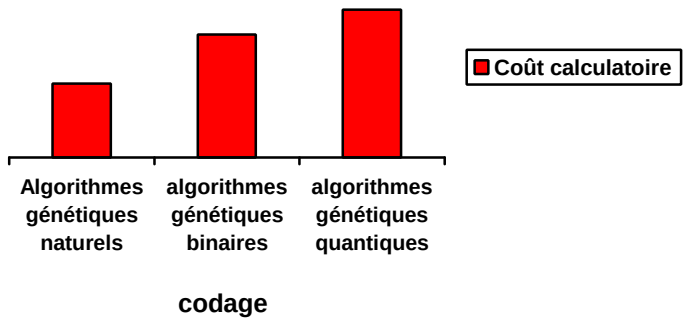


Figure 65. Graphe 14 : coût de calcul vis-à-vis le codage des individus.

D'après le (Graphe 13) on remarque que l'évolution génétique naturelle est de loin la meilleure que celles binaires et quantiques. Ces dernières n'arrivent même pas à atteindre une qualité semblable au mauvais recalage par algorithmes génétiques naturels.

Le décodage des individus binaires, et l'observation des individus quantiques pénalisent encore les schémas d'évolution non naturels, le coût calculatoire de ces derniers est non négligeable (voir Graphe 14).

Durant cette partie nous avons essayé d'identifier les bonnes valeurs des paramètres génétiques, afin d'améliorer les performances du processus de recalage premièrement en terme de qualité et deuxièmement en temps de réponse.

Concernant les opérateurs génétiques de reproductions, et commençons par la sélection élitiste qui réduit la qualité de recalage suivant une convergence prématurée, les schémas probabilistes (par roue de fortune et par rang de classement) augmentent la qualité de recalage d'une façon remarquable. Le croisement des meilleur/mauvais retard encore la convergence de l'algorithme et augmente ainsi le nombre d'individus à traiter donc la probabilité d'améliorer la qualité de recalage relativement au croisement meilleur/meilleur qui conduit le processus de recalage vers une seule direction, la direction des meilleurs individus. Comme nous l'avons vu, le tri des populations entraîne un surcoût non négligeable ce qui favorisera (peut être !) les opérateurs d'évolution par tournois même s'ils diminuent la qualité de recalage.

Concernant le choix de codage, on adopte sans la moindre hésitation la codification naturelle, elle assure les meilleures performances (qualité et temps de calcul).

Validation du choix des paramètres sur d'autres problèmes dont on connaît la meilleure solution

A travers la première partie nous avons essayé de régler les paramètres génétiques, pour s'en sortir avec plusieurs configurations gagnantes. Mais les tests de réglage ont été effectués pour résoudre le même problème avec les mêmes images et pour trouver la même transformation.

Est-ce que notre choix s'applique à d'autres problèmes ? Pour répondre à cette question on propose de tester notre solution sur d'autres problèmes, plus précisément d'autres images. On va choisir 6 images quelconques et de différentes résolutions qui seront les images consignes, pour chaque image on génère une image transformée aléatoirement qui sera l'image à recaler. On essai de résoudre chaque problème, constitué de chaque couple d'images, par plusieurs configurations génétiques. Les résultats présents dans le tableau ci-dessous sont les résultats intermédiaires de nos tests, i.e. pas les meilleurs et pas les mauvais, cela pour ne pas valoriser une chance et pour ne pas pénaliser la configuration.

Rdeux= rapport de corrélation.
CT = coût total.
CE = coût d'évaluation.
CT =coût de transformation d'images.
CS = coût de calcul de similarité.
CA = coût des autres traitements.
CT = CE + CA = (CT+CS) + CA

Tableau 11. Test des meilleures configurations génétiques sur des problèmes connus.

D'une façon générale on remarque qu'on peut valider notre modèle en termes de qualité de recalage, car tous les tests qu'on a faits (avec un choix judicieux des paramètres génétiques) convergeaient vers des bons résultats.

On remarque que le temps de réponse évolue vis-à-vis l'évolution de la résolution des images (premier test), et la taille de population (les derniers tests). L'impact de la résolution des images est le plus important, car même la configuration génétique la plus rapide (opérateurs par tournois binaires) n'arrive pas à diminuer le temps de réponse.

Ceci dit, que l'évaluation prend la part du lion, on peut même dire qu'elle est le coût de calcul. Le tableau indique que dans la totalité des tests, l'évaluation prend 99% du coût total en temps machine. Plus précisément, étant donnée que l'évaluation d'un individu (qui est une transformation) passe tout d'abord par le calcul de l'image transformée (par cet individu), ensuite retourne la valeur de la similarité entre l'image consigne et l'image transformée. Apparemment, le calcul de l'image transformée prend 96% du coût total de l'opération de recalage, cela est dû à la complexité des fonctions mathématiques (deux sinus, deux cosinus, 4 multiplications et deux additions) et non pas au parcours de l'image, car on parcourt l'image une deuxième fois pour calculer la similarité. Cette tâche ne consomme que 03% du coût total de recalage. Ces résultats sont précieux, notamment pour une future tentative d'accélération du processus de recalage.

VIII.5.8. Les images multimodales

Dans ce qui précède, nous avons démontré que notre solution marche, et elle est capable de recaler n'importe couple d'images dont l'une est la transformée de l'autre. Mais ce genre de problèmes est inclut dans l'ensemble des problèmes de recalage d'images monomodales. Maintenant il faut se dire, si notre prototype arrive à recaler deux images de différentes modalités ou non.

On dispose de deux approches pour recaler deux images de différentes modalités

- La première d'extraire l'information commune dans les deux images sans les détails (contours), ensuite recaler les deux images en utilisant un critère de calcul de similarité ordinaire (tel le rapport de corrélation).
- La seconde utilisation d'un critère qui calcule la similarité entre deux images d'une façon globale (par régions et non par un parcours des pixels). L'information mutuelle et l'entropie mutuelle sont des procédures de calcul de similarité globale [**SAL01**]

VIII.6. EXECUTIONS DE LA METHODE RECUIT SIMULE

VIII.6.1. Calibrage des paramètres recuit simulé

Nous avons constaté que les qualités de recalage à base de recuit simulé étaient relativement stables (après plusieurs exécutions on obtient des résultats très proches). La probabilité de convergence vers une classe de résultats (celle la plus probable) soit 1 à présent, La forme des graphes va changer.

Les paramètres de recuit simulé les plus importants sont

- Alpha (vitesse de diminution de la température)
- Le voisinage ou espace de recherche.
- Le nombre d'itérations par laps (ou étape).
- Les fonctions qui génèrent de nouveaux états voisins.
- le nombre laps (ou étapes) de refroidissement.
- La température initiale.
- Et en fin, le codage des états.

Dans ce qui suit, on va essayer de trouver les bonnes valeurs pour chaque paramètre. Cela en utilisant le problème défini dans le début de ce chapitre, et adopté pour le choix des paramètres des algorithmes génétiques.

On rappelle que la température initiale est initialisée de sorte que la probabilité d'adopter la mauvaise solution soit supérieure ou égale à 0,8.

VIII.6.2. Taille de l'espace des voisins

Dans cette étape, on va essayer de mesurer l'impact de la taille d'espace des voisins sur la qualité de recalage.

Les autres paramètres sont fixés comme suit:

Tableau 12. Tableau des paramètres

Méthode de similarité	**Rapport de corrélation**
Nombre des itérations pour initialiser T0	10
Nombre des itérations par étape	20
Nombre d'étape	20
Alpha (pour la diminution de T courant)	0,88
Solution initiale (ou par défaut)	X=0, Y=0, Alpha=0
Méthode de génération du voisin	Ajouter un vecteur suivant la loi normale

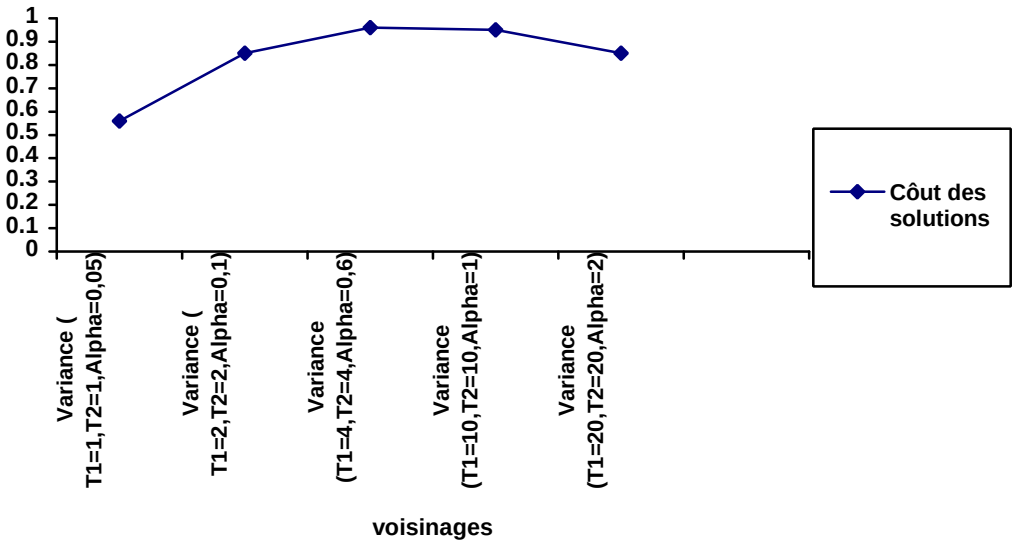


Figure 66. Graphe 15 représentant l'impact des bornes du voisinage sur les résultats de recuit simule.

Ce qu'on doit dire est qu'une taille d'un espace de voisinage large permet de grands déplacements donc conduit le processus d'exploration de visiter des états (ou des régions) dont la distance est trop grande, cela empêche le processus de recalage de se concentrer dans une éventuelle région intéressante, donc conduire le processus de recalage vers une solution généralement non acceptable comme le montre le graphe, par contre si on restreint la taille du voisinage pour garantir des tous petits déplacements alors le processus aura du mal à suivre la progression vers la bonne solution avant le refroidissement de la température, alors il va converger vers une solution de qualité moyenne. En fin un voisinage ni trop grand ni trop petit garantit des déplacements moyens, alors une meilleure exploration de l'espace de recherche, et une meilleure intensification dans le cas où on trouve une région prometteuse.

Pour conclure on peut affirmer que la taille du voisinage doit être un compromis entre une meilleure exploration de l'espace de recherche et une meilleure intensification dans le cas où on trouve une région suspecte d'être la résidence de la solution optimale.

VIII.6.3. Rapidité de refroidissement (Alpha)

Pour ce test, nous allons prendre différentes valeurs d'Alpha comprises entre [0,5 et 1] pour mesurer son impact sur la qualité des solutions.

Les autres paramètres sont fixés

Méthode de similarité	**Rapport de corrélation**
Nombre des itérations pour initialiser T0	10
Variance de T1	6
Variance de T2	6
Variance d'Alpha	0,2
Nombre des itérations par étape	20
Nombre de laps	20
Solution initiale (ou par défaut)	X=0, Y=0, Alpha=0
Méthode de génération du voisin	Ajouter un vecteur suivant la loi normale

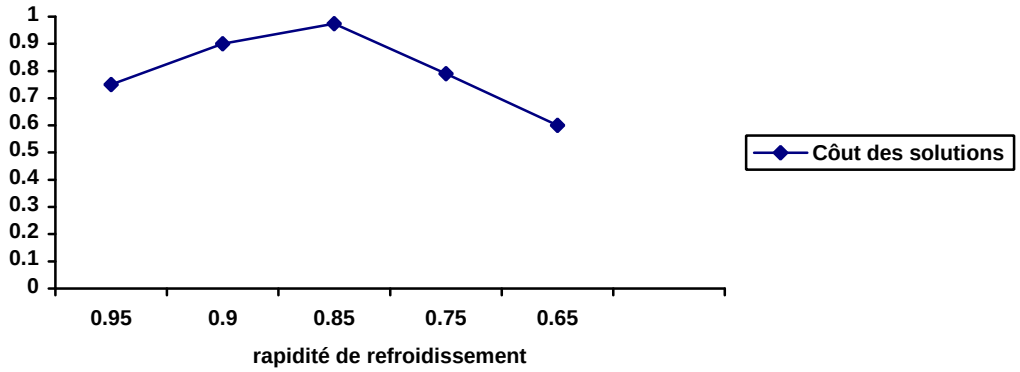


Figure 67. Graphe 16 représentant l'impact de Alpha sur les résultats de recuit simulé.

On remarque que Alpha joue un rôle important, La valeur Alpha =0,88 garantit un refroidissement progressif de la température, il n'est ni rapide comme pour Alpha=0,65 qui aboutit à une convergence systématique vers un optimal local, ni lent pour Alpha=0,95 qui demande plus de temps pour converger vers des solutions de très bonne qualité.

VIII.6.4. Le nombre des étapes

Rappelons que le nombre de laps (ou étape) c'est le nombre de fois à refroidir la température.

Dans cette partie on va essayer de mesurer l'impact réel de ce paramètre sur la qualité des solutions et le temps de calcul du recalage.

Les autres paramètres sont fixés

Méthode de similarité	**Rapport de corrélation**
Nombre d'itérations pour initialiser T0	10
Variance de T1	6
Variance de T2	6
Variance d'Alpha	0,2
Nombre des itérations par étapes	20
Alpha (pour la diminution de T courant)	0,88
Solution initiale (ou par défaut)	X=0, Y=0, Alpha=0
Méthode de génération du voisin	Ajouter un vecteur suivant la loi normale

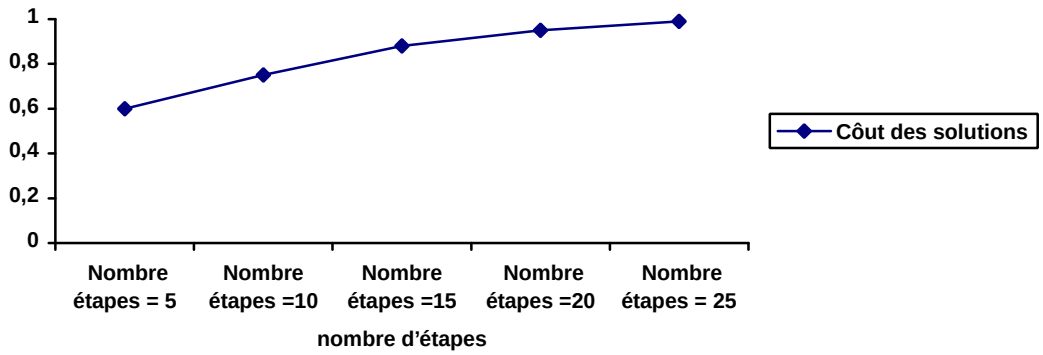


Figure 68. Graphe 17 représentant l'impact du nombre d'étapes sur les résultats de recuit simulé.

Le temps moyen de chaque test Le nombre des étapes=5 2,73 S Le nombre des étapes =10 5,24 S Le nombre des étapes =15 7,73 S Le nombre des étapes =20 10,25 S Le nombre des étapes =25 14,05 S

On remarque que plus on augmente le nombre de laps, le rendement en qualité augmente. Ceci est évident puisque le refroidissement de la température passe par plus d'étapes, se qui conduit à une convergence systématique de la méthode vers l'optimums global. Mais, à partir d'une certaine valeur (nombre d'étape=20), on remarque que le rendement se stabilise, de plus la qualité de recalage atteint le niveau très bon.

Cependant, malgré que le plus (une valeur supérieur au seuil nombre d'étape=20) ne peut pas nuire la qualité de recalage, il augmente énormément le temps de réponse de la procédure de recalage, cela est causé par le traitement inutile de l'optimisateur dans les étapes supplémentaires. Donc avec ce paramètre on tombe surtout dans un problème où il faut trouver un compromis qualité/temps de réponse.

VIII.6.5. Le nombre des itérations par étape

Rappelons que le nombre d'itérations par laps c'est le nombre d'itérations nécessaires pour atteindre le quasi-équilibre, l'état du quasi-équilibre indique que la majorité des solutions ont été visités avec la température courante.

Dans ce qui suit, on va analyser l'impacte du nombre des itérations par étape, évidement sur la qualité du recalage et le temps de calcul.

Les autres paramètres sont fixés, comme suit

Méthode de similarité	**Rapport de corrélation**
Nombre d'itérations pour initialiser T0	10
Variance de T1	6
Variance de T2	6
Variance d'Alpha	0,2
Nombre des étapes	20
Alpha (pour la diminution de T courant)	0,88
Solution initiale (ou par défaut)	X=0, Y=0, Alpha=0
Méthode de génération du voisin	Ajouter un vecteur suivant la loi normale

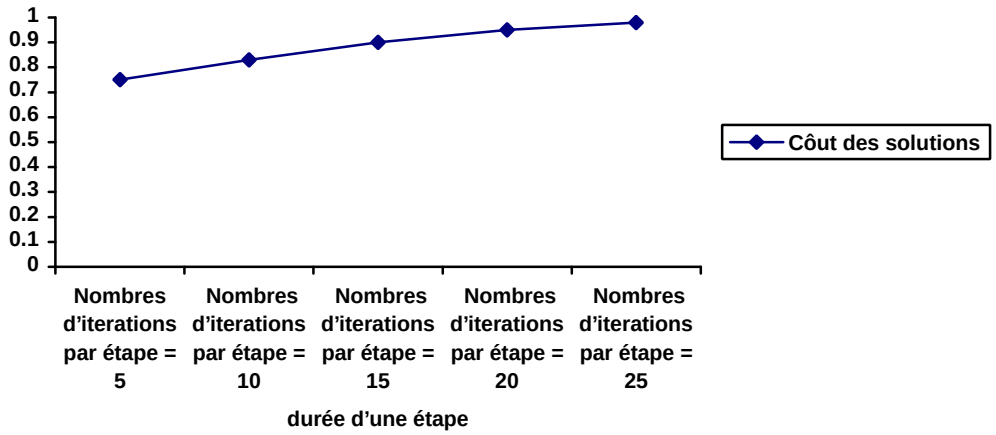


Figure 69. Graphe 18 Impact du nombre d'itérations par étape sur les résultats de recuit simulé.

Le temps moyen de chaque test
Le nombre d'itérations=5 2,48 S Le nombre d'itération=10 4,76 S Le nombre d'itération=15 6,95 S Le nombre d'itération=20 9,193 S Le nombre d'itération=25 11,35 S

On remarque que lorsqu'on augmente le nombre d'itérations par laps on assure une meilleure exploration de l'espace de recherche, c.à.d en visitant plus d'états on augmente la qualité de recalage, le graphe témoigne pour cette proposition.

Encore une fois il faut dire qu'il y a un autre coût relatif à la variation de ce paramètre qui est le temps de réponse, et qu'il faut trouver le compromis entre qualité et temps de calcul.

VIII.6.6. Fonction de voisinage

Comme déjà vu, les méthodes de génération des états voisins, sont des procédures pour créer une nouvelle configuration à partir d'une configuration courante toute en apportant une légère modification.

Dans ce paragraphe on va analyser le rôle joué par cet opérateur dans l'équation de variation de la qualité de recalage, les fonctions a analysé sont

1. Génération par mutation des paramètres.
2. Génération par addition d'un vecteur suivant la loi uniforme.
3. Génération par addition d'un vecteur suivant la loi normale.
4. Génération par changement périodique d'un élément par une valeur suivant une loi à uniforme.
5. Génération par changement périodique d'un élément par une valeur suivant une loi normale.
6. Génération indépendante de l'état courant.

Les autres paramètres sont fixés.

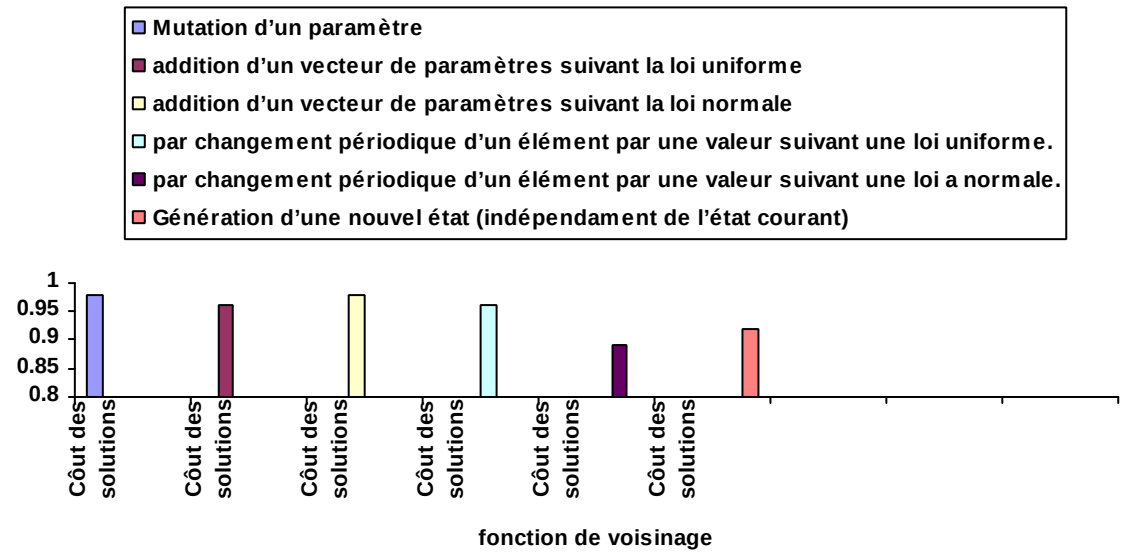


Figure 70. Graphe 19 Energie des solutions trouvées / méthodes de génération des solutions voisines.

On remarque que les différentes méthodes implémentées donnent différents résultats, et que les deux méthodes (par mutation des paramètres et par addition d'un vecteur normale) accordent les meilleurs résultats.

VIII.6.7. Type du codage des solutions

Rappelons qu'on a défini dans la partie implémentation une méthode recuite simulée avec codage binaire des solutions.

Dans ce qui suit, nous allons analyser l'impact de la codification des solutions sur le rendement de qualité de recalage et le coût de calcul exprimé en secondes.

Les autres paramètres sont fixés.

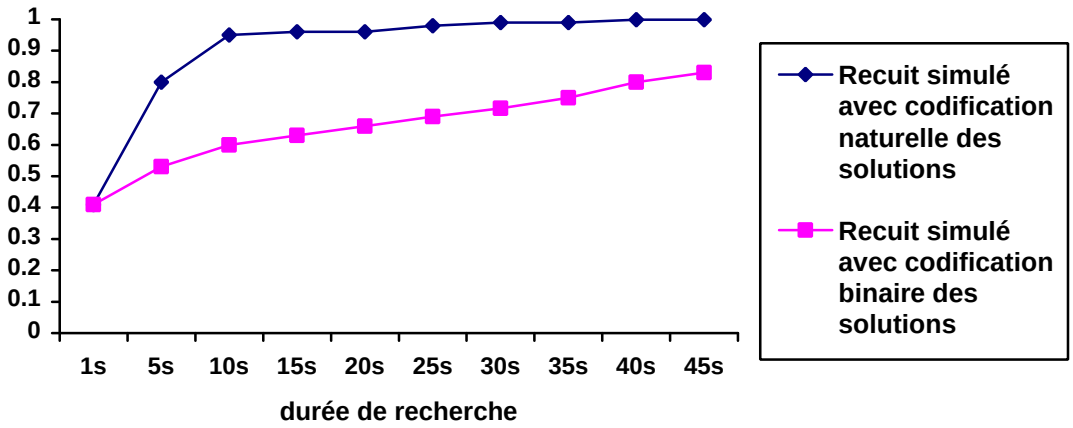


Figure 71. Graphe 20 Qualité de recalage.

On remarque que le recalage avec une codification naturelle donne de meilleurs résultats par rapport à l'autre type de codification binaire, de plus le temps total à trouver un très bon recalage est de 10s pour la codification naturelle que l'autre type de codification n'arrive même pas à le trouver qu'après 45s, a présent nous constatons que la codification binaire n'apporte pas des meilleurs résultats par rapport à la codification naturelle.

Le recuit simulé est une méthode d'optimisation globale paramétrique, ce qui nécessite l'étude de l'impact de chaque un de ces paramètres sur la qualité des résultats et le coût de calcul. Il nous a fallut garantir un paramétrage (a travers une analyse des graphes) de ces derniers qui augmentent la chance de convergence de l'optimisateur vers des résultats typiquement acceptables, avec un minimum coût de calcul relatif.

Nous allons citer maintenant chaque paramètre et choisir sa valeur convenable :

- La méthode de génération d'un nouvel état voisin par addition d'un vecteur aléatoire suivant la loi normale, cette fonction garantira la meilleure qualité des résultats de recalage.
- Espace de voisinage (espace de recherche) le voisinage = {Variance T1=6, T2=6, Alpha=0,2}.
- Alpha la valeur de diminution de T courant Alpha=0,88.
- Le nombre d'itérations par étape et le nombre d'étapes dépendront de la qualité de recalage attendue et l'espace de voisinage choisi.

VIII.6.8. Tests des paramètres sélectionnés sur d'autres problèmes pour valider le choix

Les tests de calibrage vu précédemment, en abouti à choisir des paramètres pour le recuit simulé qui donnent de meilleures résultats en un temps raisonnable, pour but de validé ces derniers, on a besoin de les analysés sur des problèmes réel. Alors, il faut se dire si ce choix est judicieux pour tous les problèmes de recalage.

Ce paragraphe est, donc, consacré pour répondre à cette question. Divers transformations ont été appliquées sur diverses images de diverses résolutions pour constituer plusieurs problèmes de recalage à résoudre.

En termes de qualité de recalage, tous les processus de recalage ont donné de bons résultats, mais le temps de réponse a été très important, ce dernier qui est emporté par le temps d'évaluation. L'évaluation constitue la partie la plus consommatrice (supérieur à 99% du coût total de recalage).

VIII.6.9. Problèmes réels

Une fois les paramètres d'atout sont identifiés, on va les appliquées directement pour résoudre des problèmes de recalage réel.

Les deux premiers tests, sont des recalages d'images monomodales inter-regions, le rapport de corrélation est utilisé pour exprimer la similarité entre images.

Les deux autres tests, sont des recalages multimodaux qui se basent sur l'entropie mutuelle comme critère de similarité.

Les résultats qui viennent d'être élaborés, sont impressionnants, la qualité est excellente de plus le temps d'évaluation est très court (relativement aux résultats des algorithmes génétiques) (pour les images de 256*256 la moyenne du temps pour le recalage est de 9,1s), avec ces tests sur des cas de recalage réel (cas des images satellitaires) on peut valider définitivement le modèle de recalage basé sur l'optimisation par recuit simulé.

VIII.7. COMPARAISON « RECUIT SIMULE VS ALGORITHMES GENETIQUES »

Nous allons entreprendre notre comparaison sur trois points

- La robustesse de chaque méthode.
- La qualité des résultats.
- Le coût de calcul.

Rappelons que la robustesse d'une méthode paramétrique reflète sa capacité de garder les mêmes performances suite à une légère modification de ses paramètres. La robustesse signifie, aussi, la facilitée de paramétrage d'une méthode.

VIII.7.1. Réglage des paramètres

- **Les AGs** un algorithme génétique a peu de paramètres à régler, l'opérateur de sélection et celui de croisement et la taille de la population. Le réglage des paramètres d'un algorithme génétique pour résoudre un problème de recalage est facile car le seul paramètre qui présente le problème qualité/temps de réponse est la taille de population. les algorithmes génétiques sont plus robustes.

- Contrairement pour le **recuit simulé** qui possède trop de paramètres à régler et qui présentent le problème qualité/temps de réponse (le nombre de laps, le nombre d'itérations par laps), cela d'une part, et d'autre part il possède des paramètres qu'il faut les régler dans un intervalle bien précis pour ne pas nuire à la qualité de recalage. Le recuit simulé et beaucoup moins robuste (relativement aux algorithmes génétiques).

- **Qualité de recalage** on peut dire que les deux modèles délivrent des solutions de même qualité, avec une meilleure performance du modèle d'optimisation par recuit simulé.
- **Temps de réponse** le recuit simulé est de loin plus rapide que les algorithmes génétiques.

VIII.8. SYNTHESE DES RESULTATS

Dans ce paragraphe on va essayer de synthétiser l'ensemble des résultats apportés par les différentes méthodes exécutées, les algorithmes génétiques (naturelles, binaires, quantiques) et le recuit simulé (naturel et binaire).

Le tableau résume le coût de calcul de la méthode la plus légère qui garantit une convergence vers des solutions acceptables :

	codage	500*500	640*768	1024*768	2024*2000
Algorithmes génétiques	Naturel	**103.54s**	**150.32s**	**300.81s**	**503.4s**
	Binaire	**125.12s**	**187.02s**	**340.15s**	**578.41s**
	Quantique	**132.91s**	**221.1s**	**396.61s**	**644.01s**
Recuit simulé	Naturel	**46.5s**	**73.2s**	**106.03s**	**245.78s**
	binaire	**67.02s**	**89.55s**	**145.58s**	**293.33s**

Tableau 13. Coût de calcul de chaque méthode.

Voici quelques statistiques concernant la consommation (en temps de calcul) de quelques parties de l'algorithme d'optimisation :

a. Algorithmes génétiques

codage	évaluation		Autres
	Transformation	similarité	
Naturel	**96%**	**03%**	**01%**
Binaire	**88%**	**02%**	**10% (décodage)**
Quantique	**88%**	**02%**	**10% (observation)**

Tableau 14. Consommation de chaque partie d'un algorithme génétique.

b. Recuit simulé

codage	évaluation		Génération des états voisins
	Transformation	similarité	
Naturel	**96,99%**	**03%**	**0.01%**
Binaire	**81%**	**02%**	**17% (décodage)**

Tableau 15. Consommation de chaque partie d'une méthode recuit simulé.

VIII.9. AUTRES TESTS

Notre solution apportée au problème de recalage se base entièrement sur les métaheuristiques, or ces derniers sont des méthodes aléatoires guidées qui s'appuient sur leurs paramétrages interne pour garantir des solutions de qualités avec des meilleurs délais de temps de réponse, par conséquence ces paramètres critiques valident leurs choix au programme par le fait qu'ils prouvent leurs performances par le biais du temps de réponse ainsi que la qualités d'optimisation.

Dans une partie antérieure nous avons procédé a amélioré ces paramètres en jouant sur l'impacte de chacun d'eux pour qu'à la fin en arrivé a bien calibré l'optimisateur.

Dans ce qui suit nous allons pencher sur des cas réel pour encore prouvé et validé notre choix de paramètres, l'accent est mis sur les images que nous avons choisies bien que leurs variétés (leurs sources) ainsi que leurs thématiques sont assez limités.

Les images dont on dispose, nous allons les utiliser pour la validation, elles nous proviennent du satellite Landsat7, les images représentent des projections sur la ville d'Alger Centre (Nord de l'Algérie) et ses environs d'une seule bande (canal) des spectres d'observation du satellite.

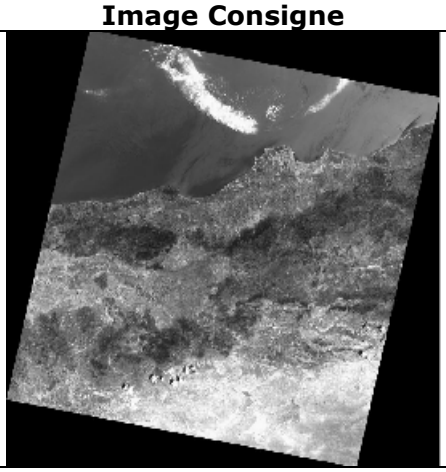
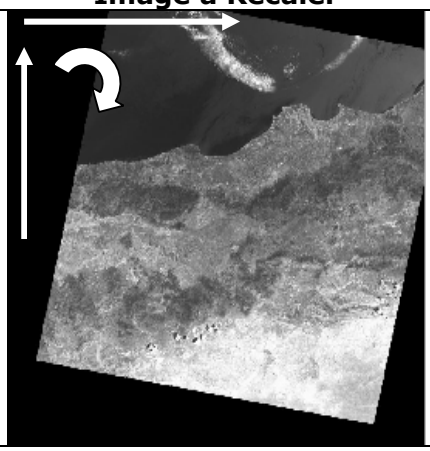
Caractéristiques des images

Caractéristiques	Informations
Source	Landsat 7
Format	TIFF
Longueur d'onde	0,45 - 0,52 (bleu)
Echelle	30 m/pixel
Résolution	14736*14072 pixels

Tableau 16. Tableau des paramètres des images.

VIII.10. TESTS & RESULTATS

VIII.10.1. Exemple 1

Image Consigne	Image à Recaler
	

La différence entre l'image consigne et l'image à recaler se resume au transformation rigide
L'optimisateur choisi pour ce problème est **AGs**
Les résultats
R_deux=0.94
Nombre de générations=14.
Coût de calcul=67.401s
99% pour le calcul des fitness = (96% pour le calcul des images transformées et 3% pour le calcul des fitness)
1% pour les opérateurs génétiques.
Fusion parfaite,

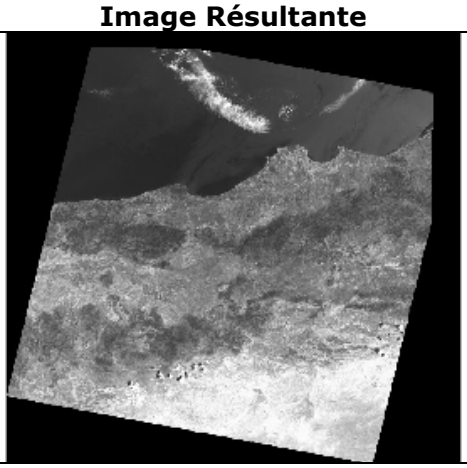
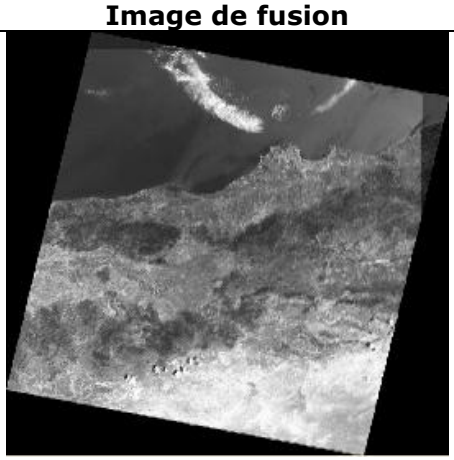
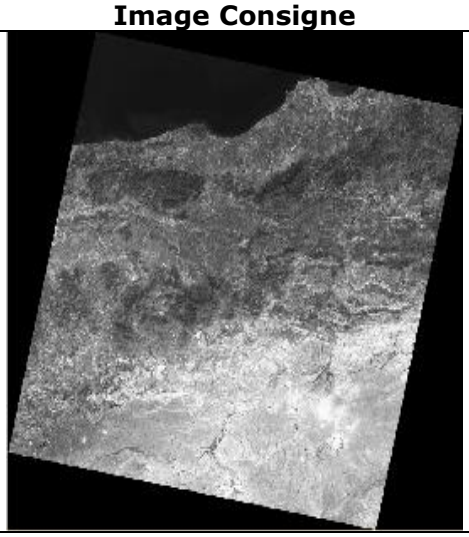
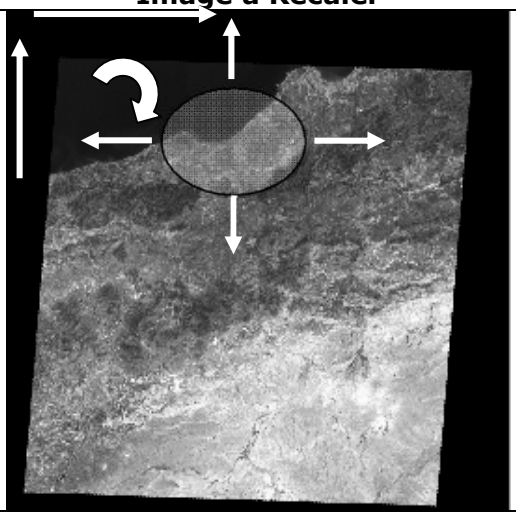
Image Résultante	Image de fusion
	

Figure 72. *Recalage d'images satellitaires rigides par les métaheuristiques.*

VIII.10.2. Exemple 2

Image Consigne	Image à Recaler
	

La différence entre l'image consigne et l'image à recaler se resume au transformation déformable
L'optimisateur choisi pour ce problème est RS
Les résultats
R_deux=0.91
Coût de calcul=33.792s
99% pour le calcul des fitness = (96% pour le calcul des images transformées et 3% pour le calcul des fitness)
1% pour l'exploration de l'espace de recherche.
Fusion moyenne,

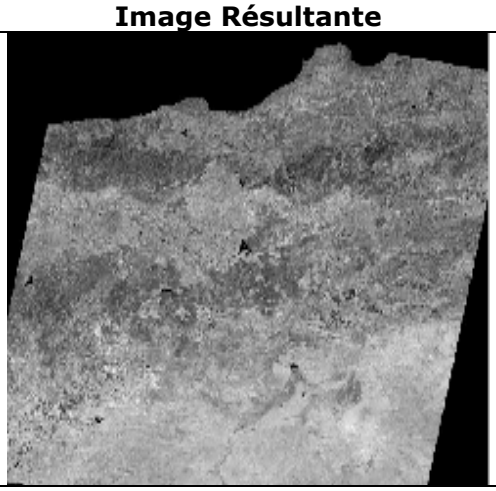
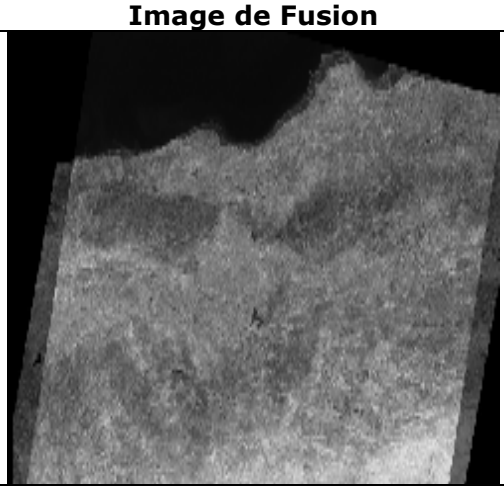
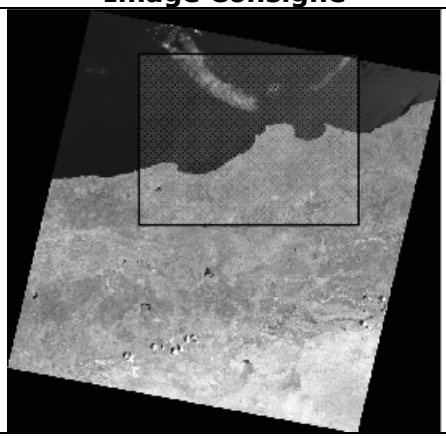
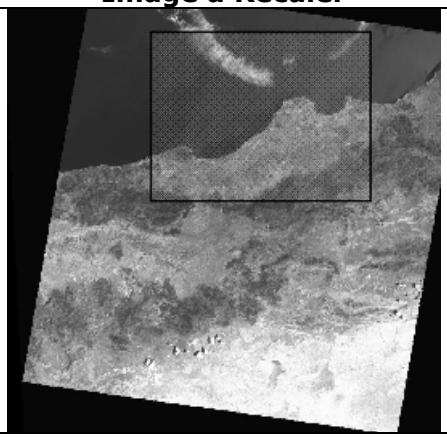
Image Résultante	Image de Fusion
	

Figure 73. *Recalage d'images satellitaires déformables par les métaheuristiques.*

VIII.10.3. Exemple 3

Image Consigne	Image à Recaler
	

La différence entre l'image consigne et l'image à recaler est inconnu.
L'optimisateur choisi pour ce problème est RS
Nous utilisons les zones d'intêts pour la résolution de ce problème.
Les résultats
R_deux=0.89
Coût de calcul=9.867s
99% pour le calcul des fitness = (96% pour le calcul des images transformées et 3% pour le calcul des fitness)
1% pour l'exploration de l'espace de recherche.
Fusion parfait au sein de la zone d'intérêt.

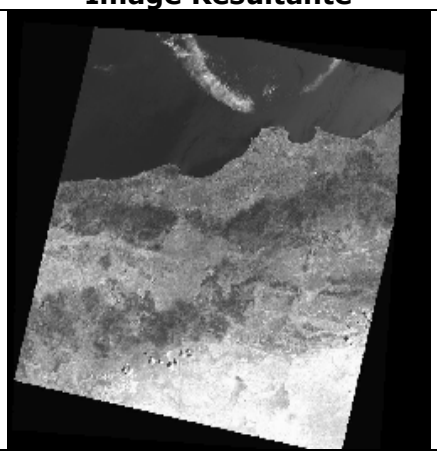
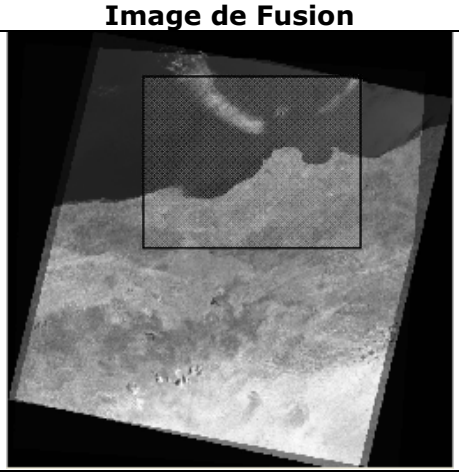
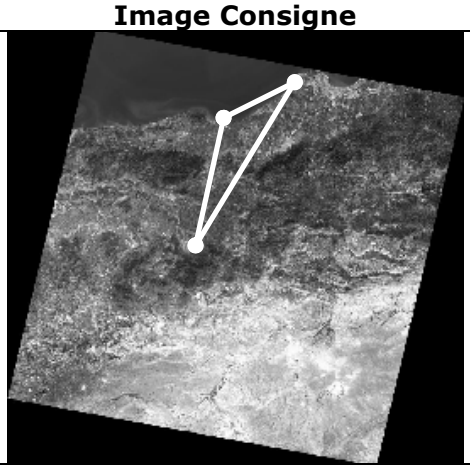
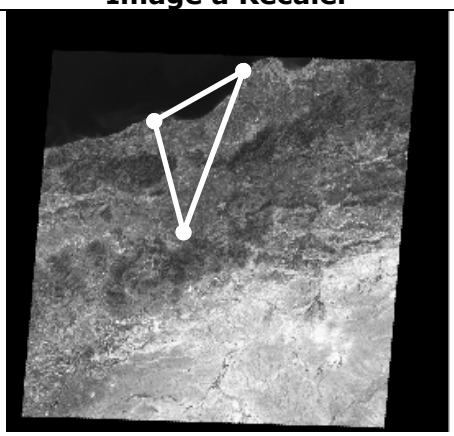
Image Résultante	Image de Fusion
	

Figure 74. *Recalage d'images satellitaires par les métaheuristiques dune zone d'intérêt.*

VIII.10.4. Exemple 4

Image Consigne	Image à Recaler
	

La différence entre l'image consigne et l'image à recaler est inconnu.
Nous utilisons les points de références pour la résolution du problème.
Les résultats
R_deux=0.83
Coût de calcul=67s
Fusion moyenne.

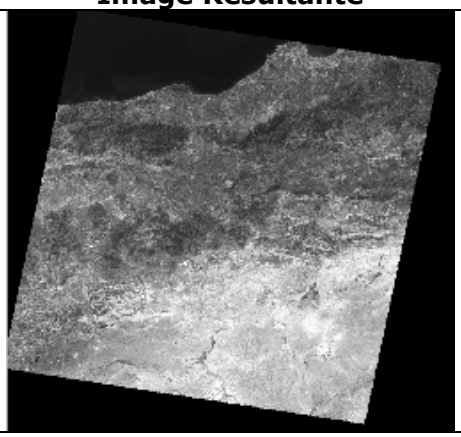
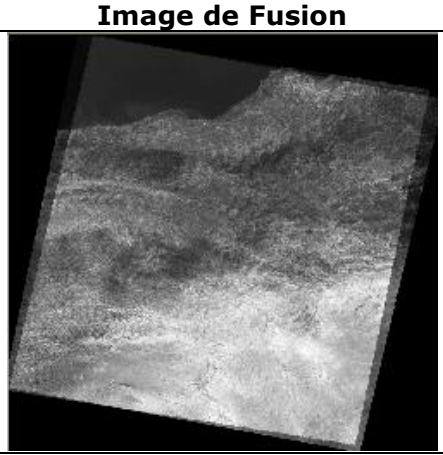
Image Résultante	Image de Fusion
	

Figure 75. *Recalage d'images satellitaires par les points de références (amers).*

VIII.11. CONCLUSION

A travers ce chapitre nous avons essayé de calibrer les paramètres de chaque méthode d'optimisation afin de garantir une convergence de bonne qualité. Une fois le choix des valeurs est fait, nous avons essayé de résoudre quelques problèmes de recalage réels. Les résultats apportés peuvent valider définitivement (théorique et expérimentale) le modèle de recalage à base de métaheuristiques.

En l'occurrence de ces tests, les résultats que nous venons d'avoir, nous conduisent à dire que les paramètres fixé préalablement, donnent des résultats appréciables et même positifs du fait qu'ils ont pu résoudre le problème en question.

Par contre la nouveauté dans ces derniers tests, est l'introduction du recalage semi automatique qui nécessite une intervention de l'utilisateur pour délimiter les zones de ressemblances (zone d'intérêt) ou fixer les points de références (points d'amers), nous avons vu que le semi automatique donnent des bons résultats bien, ce n'est qu'un point de départ pour lancer le raclage automatique et pour affiner le traitement.

Conclusion & perspectives

CONCLUSION GENERALE & PERSPECTIVES

Le recalage des images est d'une nécessité importante surtout dans le domaine de la télédétection, vu les applications de diagnostiques et thématiques très nombreuses de haute importance pour les géomaticiens, les géophysiciens et les géographes qui peuvent être facilitées considérablement moyennant le recalage. Ce dernier est aussi une tache difficile relativement à la quantité importante des informations à traiter d'une part, et la nature de paysage de la fonction à optimiser (qui est fortement non linéaire) d'autre part. Et entre ces deux points les recherches se sont orientées essentiellement vers la modélisation qui répond au mieux aux attentes des cadres de la télédétection. Et malgré les efforts, apparemment il n'existe pas une approche permettant de répondre aux attentes, qui sont toutes simples.Une bonne déformation en un temps particulièrement limité.

Par exemple les modèles élastiques non linéaires et fluides produisent des résultats de très bonnes qualités, mais le coût calculatoire d'optimisation de la déformation est trop élevé. Donc ces deux approches restent prohibées jusqu'à nouvel ordre.

Le recalage d'image étant la maximisation de la ressemblance d'image, il devient ainsi l'un des problèmes d'optimisation qui nécessite une méthode d'optimisation adéquate.

Les métaheuristiques étant les meilleures méthodes d'optimisation, grâce à leurs stratégies intelligentes de recherche, peuvent être, théoriquement, adéquats au problème de recalage. C'est pour cela que nous avons étudié l'adéquation théorique de deux métaheuristiques (algorithmes génétiques et recuit simulé) avec tous les schémas de codage possibles (naturel, binaire ou quantique).

Pour valider le modèle (recalage par métaheuristiques), nous avons mis au point «RIS-MH» une plateforme pour le Recalage d'Images Satellitaires basée sur Métaheuristiques. «RIS-MH» avec sa grande flexibilité, permet l'intégration de n'importe quelle méthode d'optimisation pour le recalage, et grâce à la séparation des contextes « optimisation/recalage » «RIS-MH» augmentera également la réutilisation de ses bibliothèques.

Le module d'interface utilisateur de «RIS-MH», permet le pilotage et le contrôle de la totalité des paramètres des différentes entités : problèmes de recalage, configurations génétiques, configurations recuit, type de codage, critères de similarités, etc. Cela nous donne l'aisance de faire des tests significatifs : un choix optimal des paramètres de chaque méthode, des comparaisons selon la performance entre les différentes méthodes, et surtout les résultats précieux qui ont été tirés.

La robustesse de la méthode algorithmes génétiques au problème de recalage est meilleure que celle de la méthode recuit simulé, mais la performance de la première n'était pas à la hauteur de la seconde. Le recuit simulé nous a accordé les mêmes qualités de recalage dans un temps très réduit relativement en temps consommé par les processus basés sur algorithmes génétiques.

Nous avons constaté, que l'évaluation était la partie la plus consommatrice dans les deux schémas d'optimisation, ce qui nous a guidé vers un schéma d'accélération basé sur la minimisation de la quantité d'informations à traiter. Le schéma accéléré, basé sur le recalage d'images zoomées, n'a pas accordé de bonnes solutions au problème de recalage, mais il a permit de trouver une solution relativement proche de l'optimale. Cette solution a été utilisée comme un bon point de départ pour une autre méthode. Et trouver un bon point de départ induit à la rapidité de convergence, et la diminution considérable du temps de calcul. A la fin, nous avons proposé une méthode de recalage automatique basé sur un schéma accéléré des méthodes d'optimisation dont les paramètres optimaux ont été fixés sur les valeurs identifiées durant l'étape du choix des paramètres optimaux.

Il faut dire, aussi, que le modèle de recalage (rigide, déformable) basé sur métaheuristiques peut être validé définitivement, car il a été validé théoriquement et expérimentalement. Cela encourage la continuité dans ce chemin, dans le sens d'adopter ce modèle pour d'autres types de transformations dont le degré de liberté et le nombre de variables à optimiser sont grands.

Pour conclure, durant ce projet, nous avons réalisé une plateforme et une bibliothèque ouverte à d'autres méthodes pour le recalage d'images par des métaheuristiques (les algorithmes génétiques et le recuit simulé).

LISTE DES FIGURES

Figure 1. Les étapes de la télédétection. 10
Figure 2. Le spectre électromagnétique 11
Figure 3. La réflexion terrestre. 12
Figure 4. capteur passif 12
Figure 5. Capteur actif. 13
Figure 6. Image multi spectrale 13
Figure 7. Les images numériques 16
Figure 8. Une image et son histogramme 17
Figure 9. Voisinage (en rouge) d'un pixel (en bleue). 17
Figure 10. Régions délimitées par des contours (en rouge). 18
Figure 11. Entrées et sorties de la procédure de recalage. 19
Figure 12. approches possibles pour réaliser un alignement d'images. 21
Figure 13. Les effets des différentes transformations. 24
Figure 14. : hiérarchie dans un algorithme génétique. 38
Figure 15. Principe de la sélection par roue de la fortune [MAT02]. 41
Figure 16. Croisement simple. 43
Figure 17. Croisement multiple de niveau 4. 43
Figure 18. Mutation en codage binaire. 44
Figure 19. Organigramme de déroulement d'un Algorithme génétique. 46
Figure 20. Etats physiques de la matière. 47
Figure 21. Parcours de l'espace de recherche avec le recuit simulé. 48
Figure 22. Schéma fonctionnel de l'opérateur Zoom out. 59
Figure 23. Exemples d'images Zoomées. 59
Figure 24. Ecarts maximals entre similarité d'images complètes et celle d'images zoomées. 60
Figure 25. Graphe 1 adaptation des translations. 61
Figure 26. Graphe 2 : accélération théorique et celle apportée par un recalage entre images zoomées. 61
Figure 27. Schéma fonctionnel de la procédure de recalage automatique. 62
Figure 28. *Schéma d'un calculateur SISD en cascade* 69
Figure 29. *Schéma d'un calculateur SIMD* 69
Figure 30. *Schéma d'un calculateur MIMD* 70
Figure 31. Schéma fonctionnel naturel d'un processus de recalage. 78
Figure 32. Entrées et sorties d'une méthode d'optimisation. 79
Figure 33. Recalage par métaheuristiques. 79
Figure 34. Représentation naturelle d'un individu (transformation). 80
Figure 35. Schéma général (avec choix des opérateurs) de l'évolution génétique. 86
Figure 36. Etape et choix pour une étape. 86
Figure 37. Individu en codage binaire. 87
Figure 38. Individus quantiques. 90
Figure 39. Structure d'un état. 94
Figure 40. Optimisation par recuit simulé 98
Figure 41. Clé pour le recuit simulé. 98
Figure 42. Etat en codage binaire 99
Figure 43. Séparation (sémantique) des modules de la plateforme «RIS-MH». 100
Figure 44. Les modules principaux du noyau élémentaire «RIS-MH». 101

Figure 45. Graphe d'héritage, une solution peut etre binaire quantique ou encore naturelle. 104
Figure 46. Graphe d'héritage, un individu est une solution et un algorithme génétique est un optimisateur. 107
Figure 47. Une population et un bassin génétique sont des ensembles d'individus. 108
Figure 48. Héritage entre les types d'algorithmes génétiques. 109
Figure 49. Graphe d'héritage un état est une solution, recuit simulé est une méthode d'optimisation. 109
Figure 50. Recuit simulé binaire et naturel, et les types des états qu'ils manipulent. 110
Figure 51. Image 1 : problème à résoudre, pour le choix des bonnes configurations génétiques. 120
Figure 52. Le rapport de corrélation un bon indicateur de la qualité de recalage. 122
Figure 53. Graphe 3: qualité de recalage vis-à-vis la taille de l'espace de recherche. 123
Figure 54. Graphe 4 : qualité de recalage vis-à-vis la variation de la taille des populations 124
Figure 55. Graphe 5 : Temps de réponse en fonction de la taille de la population. 125
Figure 56. Graphe 6 l'impact des opérateurs de sélection sur la qualité de convergence. 126
Figure 57. Graphe 7 Coût des calculs en fonction des sélections. 126
Figure 58. Graphe 8 : croisement meilleur/mauvais et convergence. 128
Figure 59. Graphe 9 : qualité de convergence de recalage basé sur une évolution purement aléatoire. 129
Figure 60. Graphe 10 : coût des calculs en fonction de la taille des populations. 129
Figure 61. Graphe 11 : l'évolution aléatoire est plus rapide à l'évolution non purement stochastique. 130
Figure 62. Graphe 12 : l'impact du choix du critère de similarité sur la qualité de recalage. 131
Figure 63. Graphe 13 : qualité de recalage vis-à-vis le codage des individus. 132
Figure 64. Graphe 14 : coût de calcul vis-à-vis le codage des individus. 132
Figure 65. Graphe 15 représentant l'impact des bornes du voisinage sur les résultats de recuit simule. 136
Figure 66. Graphe 16 représentant l'impact de Alpha sur les résultats de recuit simulé. 137
Figure 67. Graphe 17 représentant l'impact du nombre d'étapes sur les résultats de recuit simulé. 138
Figure 68. Graphe 18 Impact du nombre d'itérations par étape sur les résultats de recuit simulé. 139
Figure 69. Graphe 19 Energie des solutions trouvées / méthodes de génération des solutions voisines 140
Figure 70. Graphe 20 Qualité de recalage. 141
Figure 71. *Recalage d'images satellitaires rigides par les métaheuristiques.* 146
Figure 72. *Recalage d'images satellitaires déformables par les métaheuristiques.* 147
Figure 73. *Recalage d'images satellitaires par les métaheuristiques dune zone d'intérêt.* 148
Figure 74. *Recalage d'images satellitaires par les points de références (amers).* 149
Figure 75. Codage des individus quantiques. 165
Figure 76. Observation quantique. 165
Figure 77. Croisement quantique. 166
Figure 78. Mutation quantique. 166

LISTE DES TABLEAUX

Tableau 1. Bandes TM de Landsat. 15
Tableau 2. Bandes de SPOT. 15
Tableau 3. Les Fonction de décroissance de températures les plus utilisées. 51
Tableau 4. Comparaison entre les temps de réponse avant et après l'accélération. 63
Tableau 5. Table de rotation qauntique. 92
Tableau 6. Tableau comparatif des compilateurs utilisés 101
Tableau 7. Adapter les taux de similarités dans un context de minimisation. 103
Tableau 8. Paramètres tests opérateur de croisement. 127
Tableau 9. Paramètres tests tournois binaires. 128
Tableau 10. Paramètres tests critère de similarité. 131
Tableau 11. Test des meilleures configurations génétiques sur des problèmes connus. 133
Tableau 12. Tableau des paramètres 135
Tableau 13. Coût de calcul de chaque méthode. 144
Tableau 14. Consommation de chaque partie d'un algorithme génétique. 144
Tableau 15. Consommation de chaque partie d'une méthode recuit simulé. 144
Tableau 16. Tableau des paramètres des images. 145

LISTE DES ALGORITHMES

Algorithme 1. Forme générale d'un algorithme évolutionniste 38
Algorithme 2. Adaptation d'un individu. 81
Algorithme 3. Transformation rigide d'une image. 81
Algorithme 4. Evaluation des individus d'une population. 82
Algorithme 5. Sélection élitiste. 82
Algorithme 6. Sélection probabiliste. 82
Algorithme 7. Sélection par tournois binaires. 83
Algorithme 8. Croisement des individus. 83
Algorithme 9. Opérateur de croisement (quels sont les individus à croiser). 84
Algorithme 10. Opérateur de mutation. 84
Algorithme 11. Opérateur de mutation de la méthode stratégies d'évolution. 85
Algorithme 12. Nombre de bits nécessaire pour coder les solutions d'un espace de recherche. 87
Algorithme 13. Décodage des chaînes binaires. 88
Algorithme 14. Croisement par échange des blocs binaires. 88
Algorithme 15. Mutation par inversion de bits binaires. 89
Algorithme 16. Evolution génétique binaire pour résoudre le problème de recalage. 89
Algorithme 17. Opérateur d'observation quantique. 90
Algorithme 18. Décodage d'un individu quantique. 91
Algorithme 19. Croisement par échange de blocs quantiques. 91
Algorithme 20. Mutation quantique d'un individu. 92
Algorithme 21. Rotation quantique. 93
Algorithme 22. Evolution génétique quantique pour le recalage d'images. 93
Algorithme 23. Initialisation de la température 95
Algorithme 24. Acceptation des états. 95
Algorithme 25. Fin de recherche avec un nombre maximal d'étapes 96
Algorithme 26. Fin de recherche avec une température minimale. 96
Algorithme 27. Fin de recherche avec le taux des états acceptés 96
Algorithme 28. Fin de recherche avec une qualité satisfaisante. 97
Algorithme 29. Dans «RIS-MH» c'est le problème à résoudre qui évalue une solution. 103
Algorithme 30. Pseudo-code d'un algorithme évolutif quantique. 167

LISTE DES EQUATIONS

Equation 1. Fonction objective. 19
Equation 2. Formule d'une transformation rigide. 25
Equation 3. Formule d'une transformation affine. 25
Equation 4. Formule de l'erreur quadratique. 26
Equation 5. Formule de l'entropie 26
Equation 6. Formule de la covariance. 27
Equation 7. Formule du rapport de corrélation. 27
Equation 8. Formule de l'information mutuelle 27
Equation 9. Formule de l'entropie mutuelle. 28
Equation 10. Critère de similarité. 30
Equation 11. Critère de régularité. 31
Equation 12. Erreur quadratique moyenne. 33
Equation 13. Probabilité de sélection dans le cas "roue de la fortune". 41
Equation 14. Fonction de fitness. 42
Equation 15. Distribution de Boltzmann 47
Equation 16. Probabilité d'acceptation selon le critère de Metropolis. 49
Equation 17. Schéma de température classique du recuit simulé. 53
Equation 18. Schéma de température "rapide" du recuit simulé. 54
Equation 19. Accélération relative. 160
Equation 20. Accélération globale (loi d'Amdhal). 160
Equation 21. La représentation de Dirac d'un qubit. 163
Equation 22. La représentation matricielle d'un qubit. 163
Equation 23. Circuit NOT quantique. 164
Equation 24. Rotation quantique. 164

Annexes

Accélération relative

L'accélération relative est le taux d'amélioration d'une partie d'une solution. Mathématiquement cette grandeur est strictement supérieure à 1, elle représente le rapport entre le temps de réponse sans mécanisme d'accélération et avec le mécanisme d'accélération.

$$accélération = \frac{temps\_sans\_accélération}{temps\_avec\_accélération}$$

Equation 19. Accélération relative.

Cette accélération est relative car elle ne concerne que la partie améliorée.

Accélération globale (loi d'Amdahl)

L'idée est de mesurer l'amélioration globale apportée par une accélération locale

$$accélération_{globale} = \frac{1}{(1-p)+\frac{p}{a}} \geq 1$$

avec :

$p : pourcentage\_de\_la\_partie\_améliorée.$

$a : l'accélération\_relative\_de\_la\_partie\_améliorée.$

Equation 20. Accélération globale (loi d'Amdhal).

Cette formule reflet l'amélioration apportée par une accélération de a fois d'une partie que consomme p% du temps total. Elle permet surtout de guider le choix de la partie à améliorer.

Exemple

Soit un système formé de deux parties X et Y, de la sorte que

- X consomme 80% du temps de calcul de la solution.
- On peut accélérer 2 fois la partie X et 50 fois la partie Y.
- On ne peut apporter qu'une seule accélération.

Quelle est l'amélioration à adopter, celle de la partie X et de la partie Y ?

L'amélioration de la partie **X**, entraînera une accélération globale de

$$A_X = \frac{1}{0.2 + \frac{0.8}{2}} = 1.6667$$

Celle de la partie **Y**

$$A_Y = \frac{1}{0.8 + \frac{0.2}{50}} = 1.2437.$$

On conclue qu'il ne faut jamais procéder à une amélioration d'une façon empirique. Il faut d'abord collecter les informations concernant les caractéristiques du système en question, ensuite de choisir d'une façon judicieuse les améliorations adéquates.

La loi d'Amdahl ouvre une nouvelle perspective, dans le sens d'accélération de calcul, en définissant le principe « make the commun case faster ». Ce dernier chemine le choix des améliorations pour accélérer les parties les plus coûteuses même avec un faible taux d'accélération, car cela peut être meilleur qu'apporter une accélération, dont le taux est important, à une partie moins coûteuse. Ce qui a été démontré à travers l'exemple.

Informatique quantique et algorithmes génétiques quantiques

Introduction

Le grand souci des chercheurs est d'améliorer les performances de leurs solutions informatiques. Les concepteurs des équipements matériels informatiques ont faits leurs mieux pour que la loi de Moore témoigne de leurs efforts « les performances des unités centrales doublent tous les 18 mois ». Mais apparemment, les performances sont toujours bornées à cause de la barrière technologique. Comme les mathématiciens, les informaticiens avaient besoin d'une autre vision, une autre façon de voir les choses pour casser cette barrière.

La vision la plus révolutionnaire consistait à bénéficier des notions quantiques pour accélérer les tâches informatiques, les grandes lignes de cette idée là commençaient à apparaître en 1994, l'année où les chercheurs proposaient un algorithme de factorisation de complexité linéaire basé sur les notions quantiques [DRA04], alors que l'algorithme classique le plus rapide était beaucoup plus complexe.

Dans ce chapitre on va essayer d'aborder les notions les plus importantes de l'informatique et le calcul quantique, et son adaptation à certaines méthodes d'optimisation (les algorithmes génétiques quantiques).

Informatique quantique

La mécanique quantique remettait en question toutes nos connaissances, c'est pour cela que l'intégration des notions quantiques à l'informatique va remettre en question les notions les plus élémentaires de l'informatique classique, et commençant par la plus petite unité de stockage le Bit

Le Qubit (quantum binary digit)

Le bit classique ne peut avoir qu'une seule valeur à la fois, ou bien 0 ou bien 1. Un qubit ne peut être à 0 ou à 1 seulement mais également à 0 et à 1 au même temps. Cette réalité d'être en deux positions totalement distinctes s'appelle la superposition d'état, cette dernière est modélisée par un mécanisme stochastique basé sur deux probabilités la probabilité d'être à l'état 0, et celle d'être à l'état 1.

Modéliser un qubit [REI98]

Un qubit est représenté par un couple de deux paramètres α et β dites amplitudes de probabilité car

- La probabilité d'être à l'état 0 = α^2.
- la probabilité d'être à l'état 1 = β^2.
- De la sorte que $\alpha^2 + \beta^2 = 1$.

Pour représenter un qubit, plusieurs notations étaient proposées, la représentation matricielle et celle de Dirac sont les plus utilisées

Un qubit selon la représentation de Dirac [BEL03, KUK02]

Cette représentation utilise les états absolus classiques (0 et 1) pour représenter un qubit quelconque Q dont les amplitudes de probabilité sont (α, β). On utilise le symbole |0> pour désigner l'état absolu 0, |1> pour l'état absolu 1, et |Q> pour désigner le qubit Q.

Pour mettre en évidence les paramètres quantiques du qubit |Q> en utilise l'équation suivante

$$|\mathbf{Q}\rangle = \alpha|\mathbf{0}\rangle + \beta|\mathbf{1}\rangle$$

Equation 21. La représentation de Dirac d'un qubit.

Exemples

- $|0\rangle = 1|0\rangle + 0|1\rangle$ qubit toujours à l'état 0
- $|1\rangle = 0|0\rangle + 1|1\rangle$ qubit toujours à l'état 1
- $|Q\rangle = 1/\sqrt{2}|0\rangle + 1/\sqrt{2}|1\rangle$ qubit dont les deux états sont équiprobables.

Représentation matricielle [DRA04, KUK02]

Elle consiste en une projection des paramètres d'un qubit sur un vecteur colonne

$$|Q\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

Equation 22. La représentation matricielle d'un qubit.

Exemples $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ $|Q\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$

Registre quantique [GLA01]

La concaténation de n qubit créera un registre quantique de n qubit, et puisque un qubit superpose deux états distincts, alors le registre quantique superposera les 2n états possibles [BEL03].

Circuits quantiques [TAL04]

Les circuits ou portes quantiques, permettront la manipulation et l'exploitation des informations représentées par les qubits. Une porte quantique peut être une redéfinition d'un circuit classique, ou carrément un nouveau circuit. Il y a toute une panoplie des portes quantiques, mais nous on ne s'intéresse qu'à deux, le circuit NOT et la Rotation qui sont utilisés par les algorithmes génétiques quantiques.

La porte Not [DRA04]

Cette porte consiste à inverser les amplitudes de probabilité d'un bit quantique.

$$Not\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\begin{pmatrix}\beta\\ \alpha\end{pmatrix}$$

Equation 23. Circuit NOT quantique.

Exemples

- $Not(|0\rangle)=|1\rangle,\ Not(|1\rangle)=|0\rangle$

La rotation quantique [TAL04]

Elle consiste à remplacer les valeurs des amplitudes de probabilité (α, β), par des autres (α', β'), ces dernières représentent les valeurs transformées par une rotation d'angle δ, elles sont évaluées de la façon suivante

$$\begin{pmatrix}\alpha'\\ \beta'\end{pmatrix}=\begin{pmatrix}\cos(\delta) & -\sin(\delta)\\ \sin(\delta) & \cos(\delta)\end{pmatrix}*\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

Equation 24. Rotation quantique.

Algorithmes génétiques quantiques [DRA04, TAL04]

On ne peut rien faire pour améliorer les performances de certaines tâches informatiques en suivant les approches classiques. Peut être la seule manière d'améliorer leurs performances c'est d'adopter un nouvel modèle, voir les choses autrement !

Avec la révolution de l'informatique quantique, et l'évolution de ses notions intéressantes surtout pour les problèmes d'optimisations. En effet, si on arrive vraiment à modéliser une structure quantique qui va représenter une solution à un problème d'optimisation et qui peut contenir toutes les solutions possibles à ce problème, alors on sera capable de le résoudre facilement, peut être qu'il ne sera même pas considéré comme étant un problème d'optimisation. Cela est une autre façon de voir les choses, dont l'efficacité ne peut être prouvée expérimentalement sans un ordinateur quantique.

A présent on ne peut utiliser les notions quantiques avec les méthodes d'optimisations que pour améliorer leurs performances, surtout l'efficacité de la stratégie de recherche. Cette voie engendrait les algorithmes génétiques quantiques.

A travers ce chapitre on va essayer de définir les nouveautés apportées par l'hybridation quantique/génétique et de redéfinir ce qu'il y a à redéfinir, et commençons par le codage des individus

Codage des individus [DRA04]

Dans le chapitre précédent on pouvait voir les individus à travers leurs représentations naturelles, ou à travers le codage binaire. Le codage des individus dans un algorithme génétique quantique, consiste à remplacer les bits ordinaires dans la représentation binaire par des qubits.

α(1)	α(2)	…….	α(n)
β(1)	β(2)	…….	β(n)

Où

n nombre de qubits nécessaires pour coder un individu

$\alpha_i^2 + \beta_i^2 = 1$ pour **i=1..n**

Figure 76. Codage des individus quantiques.

Redéfinition des opérateurs génétiques

Puisque les opérateurs génétiques travaillent pour modifier les structures des individus d'une population donnée, alors il est trivial qu'un changement de codage des structures entraînera le changement de toutes les définitions des opérateurs.

Dans ce qui suit, on va essayer de garder l'analogie des opérateurs et de les redéfinir pour devenir compatibles avec la nouvelle structuration.

Evaluer la fitness d'un individu quantique

Tout d'abord il faut décoder la chaîne des qubits pour extraire l'information utile, ensuite évaluer son taux d'adaptation pour un problème donnée.

Le décodage se déroule en deux grandes étapes

1. Observation de l'ensemble des qubits pour obtenir la version binaire de l'individu.
2. Décoder la chaîne binaire pour extraire la représentation naturelle de l'individu.

L'observation d'un qubit se fait par une simple comparaison

1. Si l'état 0 est plus probable alors, c'est un 0.
2. Sinon c'est un 1.

Exemple

$$\begin{pmatrix} 0.9887 & 0.4132 & 0.7955 & 0.8733 & 0.7676 \\ 0.0224 & 0.8293 & 0.3672 & 0.2373 & 0.4108 \end{pmatrix} \xrightarrow{observation} (0 \quad 1 \quad 0 \quad 0 \quad 0)$$

Figure 77. Observation quantique.

Croisement quantique [TAL04]

En s'inspirant du croisement pour un codage binaire, et puisque un bit et qubit représentent des unités bas niveaux les chercheurs proposent le croisement par échange de sous chaînes quantiques. I.e. à partir de deux individus parents P1 et P2, et en prélevant un ou plusieurs points de coupures qui vont définir des blocs, on génère deux individus enfants E1 et E2, qui recevront alternativement les blocs des deux parents.

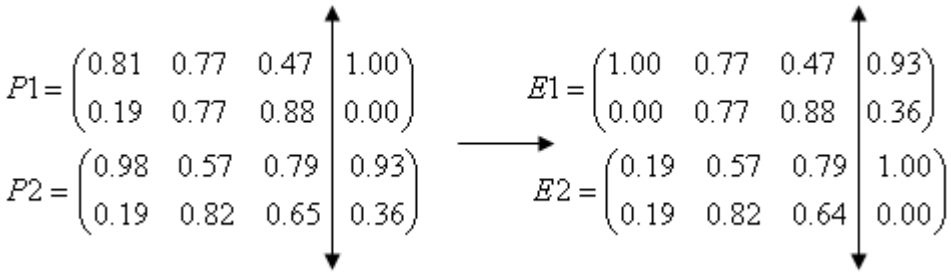


Figure 78. Croisement quantique.

La mutation quantique [TAL04]

Elle consiste à appliquer la porte Not quantique sur un ou plusieurs qubits, selon une probabilité de mutation.

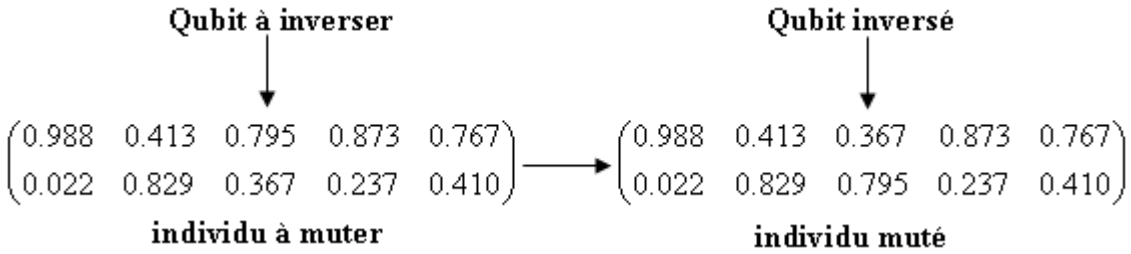


Figure 79. Mutation quantique.

Rotation quantique

Cet opérateur consiste en une rotation quantique de tous les qubit d'un individu, cet opérateur garantira une autre façon de diversification de la population se qui conduira sans doute à une meilleure exploration de l'espace de recherche, et augmentera ainsi la chance de convergence vers des bonnes régions.

Forme générale d'un algorithme génétique quantique

Une fois, tous les opérateurs définis, on peut à présent proposer le schéma général de l'évolution génétique quantique

- Initialiser la population des individus quantiques
- **Répéter**
  - **Observer** tous les individus de la population quantique pour obtenir une population ordinaire.
  - **Evaluer** la population ordinaire.
  - Appliquer les opérateurs d'évolution sur les individus quantiques
    - Sélection
    - Croisement quantique
    - Mutation quantique
  - Appliquer les circuits quantiques sur les individus quantiques
- **Jusqu'à** (critère de fin de recherche).

Algorithme 30. Pseudo-code d'un algorithme évolutif quantique.

A chaque génération on doit observer les individus pour avoir accès aux informations utiles qu'ils représentent, on peut par la suite les évaluer en calculant le taux d'adaptation de chaque individu.

En ce qui concerne l'évolution quantique, et commençant par la sélection qui n'utilise que les taux des fitness des individus pour être accomplie, ensuite croiser les individus quantiques sélectionnés selon une probabilité de croisement Pc. pour finir les opérateurs classique par une mutation avec une probabilité Pm. A la fin sont appliqués des circuits quantiques (généralement la rotation quantique) sur les individus.

Conclusion algorithmes génétiques quantiques

Grâce à la modélisation probabiliste des solutions d'un problème d'optimisation, et la caractéristique de superposition d'état d'un qubit, les algorithmes génétiques quantiques suivront une stratégie intelligente d'exploration et d'exploitation. La rotation quantique est un opérateur supplémentaire qui va augmenter, sans doute, la puissance du nouvel schéma évolutif.

On a fait le tour, à travers ce chapitre, de deux points des notions fondamentales du calcul quantique, et comment ces notions ont révolutionnées les méthodes d'optimisation, à travers l'adaptation quantique aux algorithmes génétiques.

ANNEXE II : SATELLITE IMAGES REGISTRATION BY METAHEURISTICS.

Le présent article a fait l'objet d'une communication nationale dans le cadre de la première Journée Nationale sur les Applications des Métaheuristiques (JNAM'07), le 29 mai 2007 déroulé à la Faculté d'Electronique et d'Informatique de l'U.S.T.H.B Alger.

Lahlou BENGHEZAL et Rachid OUAFI
USTHB, Faculté des Mathématiques
Département de Recherche Opérationnelle
BP 32 El Alia, 16111Bab Ezzouar, Alger.
Email : benglah_soft@hotmail.fr

Abstract. **Registration techniques in satellite image processing are used to match land structures from two or more images taken at different times to track for example the evolution of a natural phenomenon. The core of the registration process is the maximization of a cost function expressing the similarity between these images. To resolve this problem, we have tested two global optimization techniques that are genetic algorithms and simulated annealing. In this paper we show some results obtained in satellite images registration.**
Keywords: **Genetic algorithms, satellite image registration, simulated annealing.**

Résumé: **Les techniques de recalage en imagerie satellitaire sont utilisées pour faire correspondre des zones géographiques sur deux ou plusieurs images prises à des temps différents pour suivre par exemple l'évolution des phénomènes naturels. Le noyau du processus de recalage est la maximisation d'une fonction coût exprimant la similarité entre ces images et pour résoudre ce problème, nous avons testé deux méthodes d'optimisation globales que sont les algorithmes génétiques et le recuit simulé. Dans cet article, nous montrons quelques résultats obtenus dans le domaine du recalage d'images satellitaires.**
Mots clés : **Algorithmes génétiques, recuit simulé, recalage d'images satellitaires.**

I. INTRODUCTION

Satellite imaging is a vital part in several applications (military, meteorological phenomenon,). Physicians often wish to compare two or more images of the same area acquired at different times to track events. Satellite image registration provides an important aid in diagnosis of different nature.

Image registration technique [13] can be defined as a mapping between two or more images spatially and with respect to similarity. For the sake of simplicity we admit that the registration will be made between an image to register I_R and a reference image I_C.

Thus, the registration process consists in transforming the image to register I_R such as the anatomic structures expressing the same information will be superposed in the two images I_R and I_C ([10], [05]). Formally, it is the maximization of the similarity (with respect to some criteria) between the two images [07].

In practice, the registration process comes to search the transformation T that verifies the following relation:

$$Maximize\ [Similarity(I_C, T(I_R))] \qquad (1)$$

In the preceding equation, the similarity (or similarity ratio) between an image X and an image Y is expressed by Similarity(X, Y) and T(X) is the transformation of the X image by T. To solve equation (1), we have tested two global optimization techniques that are genetic algorithms and simulated annealing.

This paper is organized as follows. We define in section 2 the principles of genetic algorithms. Simulated annealing is explained in explained in section 3. Experimental results on satellite samples datasets are given in section 4. Section 5 gives conclusions.

II. GENETIC ALGORITHMS

Genetic algorithms are the results of J. Holland works [03]. Holland tries to interpret and artificially model the principles of the Darwinian evolution.

A. Description

In an analogous manner to the Darwinian evolution, applying genetic algorithms to solve an optimization problem makes evolving a population of elements (individuals) (where every individual represent a solution) to a population optimizing this problem.

Evolutionary algorithms operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals

according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

For every individual, we can measure his adaptation ratio (fitness) which is the value of the function to optimize. For the evolving process, we use a "selection" operator which chooses the best individuals of the actual population to put them in a "genetic basin". A "recombination" operator and a "mutation" operator are applied to the individuals in the "basin" to generate new individuals.

The general structure of an evolutionist genetic algorithm is the following [02]:

```
Initialization ;
Evaluation ;
Repeat
    Evolution ;
    Selection ;
    Recombination ;
    Mutation ;
    Evaluation ;
Until (stop criterion).
```

The performance and complexity of a genetic algorithm narrowly depend on the following characteristics [09]:

1. Coding: to define the intervention space of genetic operators (recombination and mutation).
2. Population size.
3. Fitness: the "quality" of a solution given by a cost function.
4. Selection mode and reproduction operators.
5. Other parameters (crossing over ratio, mutation ration… etc.)

B. Coding

In the following sections we represent an n-dimension solution by:

$$X = \{x_1, x_2, ..., x_n\} \qquad (2)$$

1. Natural coding: in this coding the characteristics of a solution are kept, an individual is represented by a vector of variables [11].
2. Binary coding: consists in coding the characteristics of a solution by a string of bits [01]. It is a blind coding.
3. Quantum coding: consists in replacing the bits of the binary coding by Qubits (quantum binary digit) [12]).

C. Genetic operators

1. Selection

The aim of this step is to choose the best individuals for the reproduction. This choice can be deterministic or probabilistic. The selection is made independently of the coding scheme. The fitness of an individual is used to accomplish the task of this operator.

2. Recombination

The recombination "crossover" operator allows the characteristics transmission from the parents to children. The crossing-over is made with a given probability. The definition of this operator is narrowly linked to the coding used.

3. Mutation

Mutation is an unary genetic operator that is characterized by the modification (with respect to a mutation probability) of the structure of an individual. Mutation operator is also linked to the coding used.

D. Stop criterion

An optimization process based on a genetic algorithm converges if the actual population is homogeneous. A population is homogeneous if all or the majority of its individuals have the same genes.

III. SIMULATED ANNEALING

Simulated annealing is a neighborhood based optimization method inspired from a technique used to have states of low energy of a material. This technique is called metallurgic or physics annealing.

A. Description

Inspired from the principles of the metallurgic annealing, Kirkpatrick [06] proposed an optimization method named simulated annealing. The energy of a material can be viewed as a cost function of the optimization problem. The different states of this material can be considered as solutions of the cost function.

The general structure of the simulated annealing algorithm is the following:

```
Initialization (X actual solution, Tactual)
   while (stop criterion non satisfied) do
         while (quasi equilibrium not reached at
         actual temperature) do
               generate a neighbor solution X'
               if (acceptation (cost(X), cost
               (X'), Tactual) then
                     update X
         end-while
         anneal (Tactual)
End-while.
```

Initially the temperature is very high i.e. all the solutions of the search space are acceptable. The simulated processes by generating and accepting a neighbor solution until quasi equilibrium reached. Annealing will follow until satisfying a stop criterion.

Being probabilistic, the simulated annealing process can accept a solution that is worst than the actual solution. With this acceptation strategy, simulated annealing avoids the trap of local optima. Accepting a neighbor solution is made by calculating an "admission" probability. The most used probability is metropolis criterion [04, 08]. If y is a neighbor solution to the actual solution x. The variation of energy is calculated f = f(y) – f(x). The solution y is accepted with respect to the probability (metropolis criterion) p(x, y) given below:

$$p(x,y) = e^{\left(\frac{f(y)-f(x)}{T}\right)} = e^{\left(\frac{-\Delta f}{T}\right)} \quad (3)$$

B. *Simulated annealing characteristics*

4. Neighborhood

The neighbor solution is calculated from a solution by applying some slight modifications.
When the algorithm progresses, a visited solution can be considered again. Nevertheless, this probability is very low id the search space is large and the function generating neighbors is well conceived.

5. Temperature

The temperature is a control parameter. When the temperature is high, new solutions can be admitted, while with low temperature global optimum is approached. The temperature must be decreased according to a function so a maximum of configurations can be tested.

C. Stop criterion

During the progression of the optimization processes, the convergence toward the optimal solution can not be determined. To overcome this difficulty, some criteria are used to appreciate the algorithm convergence. The most used criteria are:

1 Rejected solutions ratio
2 Cost variance of visited solutions
3 Minimal temperature.

IV. EXPERIMENTAL RESULTS

All the tests were made on an IBM PC compatible machine (CPU Pentium 4, 1.6 GHz and 256 Mo of RAM) under Microsoft Windows XP service pack2 operating system. The satellite images used are acquired at different times. The following table shows some registration examples. The images, scaled 30m/pixel, are provided by Landsat 7 Satellite. They describe Algiers (capital of Algeria) and neighbourhoods.

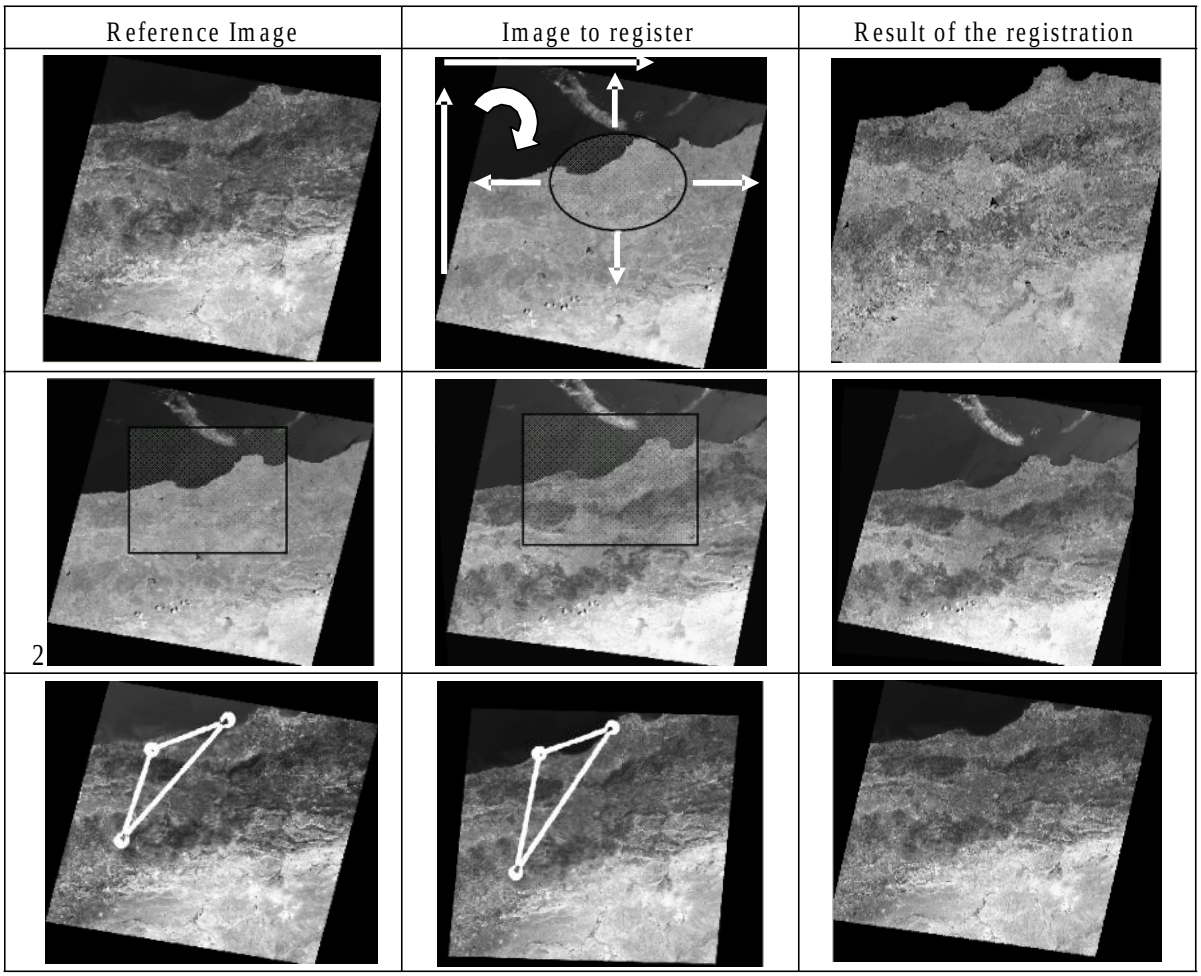


Table 1: Algiers images registration examples.

The registration tests were conducted using genetic algorithms (natural, binary and quantum coding) and simulated annealing (natural and binary coding) on the same samples. The following table summarizes the average costs of the registration process by using different optimization methods with different image sizes.

		Image size			
		500*500	640*768	1024*768	2024*2000
Genetic algorithms	Natural Coding	**103.54s**	**150.32s**	**300.81s**	**503.40s**
	Binary Coding	**125.12s**	**187.02s**	**340.15s**	**578.41s**
	Quantum Coding	**132.91s**	**221.10s**	**396.61s**	**644;01s**
Simulated annealing	Natural Coding	**46.50s**	**73.20s**	**106.03s**	**245.78s**
	Binary Coding	**67.02s**	**89.55s**	**145.58s**	**293.33s**

Table 2: Average calculating time in seconds.

$$(4)\quad R^2(I,J)=\left(\frac{COV(I,J)}{\sqrt{\sum_s (I(s)-\bar{I})^2}\times\sqrt{\sum_s (J(s)-\bar{J})_2}}\right)_2 \quad \text{for each pixel s}$$

The correlation ratio given by equation 4 was used to measure the quality of the registration process. The following table gives the correlation ration for the three registration examples shown in table 1. These results are very satisfying.

Example #	Correlation ratio
1	0.94
2	0.91
3	0.89

Table 3: Correlation ratio.

V. CONCLUSION

We have explored image registration in satellite imaging. The registration process tests were conducted on the same satellite image samples using genetic algorithms and simulated annealing.

Simulated annealing with natural coding seems to be the fastest method. Further tests must be conducted on large samples and other optimization techniques must be tried to have a more accurate synthetic view of the registration process in satellite imaging field.

BIBLIOGRAPHIE

[01] Andrey P., "Segmentation d'images par algorithmes génétiques", PhD Thesis, Paris 7 University, France, 1999.

[02] Harik G.R., Lobo F.G. & Goldberg D.E., "The compact genetic algorithm", IEEE World Congress on Computational Intelligence, pp. 523-528, May 1998.

[03] Holland J.H.., "adaptation in natural and artificialsystems", Michigan University Press, 1975.

[04] Johan D., "Métaheuristiques pour l'optimisation difficile", Eyrolles Edition, France, 2003.

[05] Juan D., Tang S., Jiang T. & Lu Z., "Intensity–based robust similarity for multimodal image registration", International Journal of Computer Mathematics, vol. 83, n° 1, Jan. 2006, pp. 49-57.

[06] Kirkpatrick S., Gelatt C.D. & Vecchi M.P., "Optimisation by simulated annealing", Science, volume 220, numéro 4598, pp 671-680, 1983.

[07] Li W. & Leung H., "A maximum likelihood approach for image registration using control point and intensity" IEEE Trans. on Image Processing, vol. 13, Issue 8, pp. 1115-1127, August 2004.

[08] Mendonca R.S. & Caloba L.P., "New simulated annealing algorithms", proceedings ISCAS '97 vol.3, pp. 1668-1671.

[09] Miramond B., "Méthodes d'optimisation pour le partitionnement logiciel/matériel de systèmes à description multi-modèles", PhD Thesis, Evry University, France, Décembre 2003.

[10] Musse O., Heitz F. & Armspatch J.P.., "Topology preserving deformable image matching using constrained hierarchical parametric models", International Conf. On Image Processing, vol. 1, pp.505-508, Sep. 2000.

[11] Salomon A., "Etude de la parallélisation de méthodes heuristiques d'optimisation combinatoire : application au recalage d'images médicales", PhD Thesis, Louis Pasteur University, Strasbourg, France, Décembre 2001.

[12] Talbi H., Dra A. & Batouche M., "A new quantum inspired genetic algorithm for solving the salesman problem", IEEE international Conf. On Industrial Technology, 8-10 December 2004, vol. 3, pp. 1192-1197.

[13] Zitova B. & Flusser J., "Image registration methods: a survey", Image and Vision Computing 21 (2003) 977–100.

Bibliographie

BIBLIOGRAPHIE

[AND99] P. Andrey « **segmentation d'images par algorithmes génétiques** » thèse doctorat, sciences appliquées, paris 7, 1999.

[BEL99] Belaji Natrajan and bruc E. Rosen « **imagine enhancmente using very fats simulated annealing** » division of cumputer science, The university of Texas at san antonio,IEEE. 1999.

[BEL03] Silva, L.; Bellon, O.R.P.; Gotardo, P.F.U.; Boyer, K.L. "**Range image registration using enhanced genetic algorithms"**Image Processing, 2003. ICIP 2003. Proceedings. 2003. International Conférence on Volume 2, 14-17 Sept. 2003 Page(s):II - 711-14 vol.3
Digital Object Identifier 10.1109/ICIP.2003.1246779

[BEN03] Benoît Rulleau « **simulation arithmétique et optimisation de la structure des zones de qualification** » thèse, laboratoire d'optimisation globale, centre d'études de la navigation Aérienne, Toulouse. 18 septembre 2001.

[BON95] Christophe Bontemps « **Principes Mathématiques et Utilisations des Algorithmes Génétiques**» 18 Novembre 1995,
URL :http://www.toulouse.inra.fr/centre/esr/CV/bontemps/WP/AlgoGene.htm

[CHA88] R.D. Chamberlain, M.N. Edelman, M.A. Franklin and E.E. Witte. « **Simulated annealing on a multiprocessor"** Computer Design: VLSI in Computers and Processors, 1988. ICCD '88. Proceedings of the 1988 IEEE International Conference on 3-5 Oct. 1988 Page(s):540 - 544

[CHR01] G.E Christensen, H.G Johnson « **Consistent image registration**"
Medical Imaging, IEEE Transactions on Volume 20, Issue 7, July 2001 Page(s):568 - 582

[CHM03] L. Chielewski and D. « **Image registration**" Medical Imaging, IEEE Transactions on Volume 22, Issue 11, Nov. 2003 Page(s):1341 - 1343

[GLA01] Glassner, A.; « **Quantum computing**. 3 » Computer Graphics and Applications, IEEE , Volume 21, Issue 6, Nov.-Dec. 2001 Page(s):72 – 82

[GOL00] Marin Golub and Domagoj Jakoviè "**A new model of global parallel genetic algorithm**" Information Technology Interfaces, 2000. ITI 2000. Proceedings of the 22nd International Conference on 13-16 June 2000 Page(s):363 - 368

[DRA04] A. DRAA, H. TALBI, M. BATOUCHE « **Une approche génétique quantique pour la fusion d'images multi-sources** » Equipe vision et infographie, Laboratoire LIRE Université Mentouri Constantine ALGERIE, 7/10/04

[HAR98] Harik, G.R.; Lobo, F.G.; Goldberg, D.E.; « **The compact genetic algorithm** » Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference on 4-9 May 1998 Page(s):523 - 528

[HOL75] J.H. Holland "**adaptation in natural and artificial systems**" university of Michigan press, 1975

[HUE03] R.H. Huesman, G.J. Klien, J.A.Kimdon, C.Kuo, S.Majumdar **« Deformable registration of multimodal data including rigid structures"**Nuclear Science, IEEE Transactions on Volume 50, Issue 3, Part 2, June 2003 Page(s):389 - 392

[ISS 04] Hazem ISSA "**Mise en correspondance des stéréoscopique par algorithmes génétiques: nouveaux codages**" thèse Doctorat Lille 1 2004.

[JON75] K. De Jong "**an analysis of the behavior of a class of genetic adaptative systems**" PhD thesis, University of Michigan, 1975

[JOH03] D. Johan et al. « **Métaheuristiques pour l'optimisation difficile** » Edition Eyrolles, 2003

[KIR83] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi « **optimization by simulated annealing** » article Science volume 200, Numéro 4598. 13 mai 1983, URL http://citeseer.ist.psu.edu/kirkpatrick83optimization.html

[KUK02] Kuk-Hyun Han; Jong-Hwan Kim; « **Quantum-inspired evolutionary algorithm for a class of combinatorial optimization**" Evolutionary Computation, IEEE Transactions on Volume 6, Issue 6, Dec. 2002 Page(s):580 - 593

[LI 04] Li, W.; Leung, H.; "**A maximum likelihood approach for image registration using control point and intensity**" Image Processing, IEEE Transactions on Volume 13, Issue 8, Aug. 2004 Page(s):1115 - 1127

[LU 91] Lu, N.-A.; Morrell, D.R.; « **VQ codebook design using improved simulated annealing algorithms**" Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on 14-17 April 1991 Page(s):673 - 676 vol. 1

[MAE03] Yoichiro Maeda and Toru Tsubouchi "**parallel genetic algorithm used fuzzy adaptative search method**" SICE annual conference in Fukui, August 4-6, 2003.

[MAE04] Yoichiro Maeda and Masahide Ishita "**fuzzy adaptative search method for parallel genetic algorithm with combined sub-populations**" proceeding of the 2004 IEEE conferences 25-29 July, 2004.

[MAT 02] Matthieu D. "**les algorithmes génétiques**" 2002.

[MEN97] Mendonca, P.R.S.; Caloba, L.P.; **« New simulated annealing algorithms »** Circuits and Systems, 1997. ISCAS '97. Proceedings of 1997 IEEE International Symposium on Volume 3, 9-12 June 1997 Page(s):1668 - 1671 vol.3

[MOI98] Le Moigne, J.; Wei Xia; Chalermwat, P.; El-Ghazawi, T.; Mareboyana, M.; Netanyahu, N.; Tilton, J.C.; Campbell, W.J.; Cromp, R.P.;First "**evaluation of automatic image registration methods**" Geoscience and Remote Sensing Symposium Proceedings, 1998. IGARSS '98. 1998 IEEE International, Volume 1, 6-10 July 1998 Page(s):315 - 317 vol.1

[OMR04] Nacer OMRANE. "**Super Resolution For Unregistered Satellite Images UNIS**" 2004. Doctor of philosophy from the University of Survey January 2004

[LAH07] Lahlou BENGHEZAL et Rachid OUAFI « **Satellite Images Registration by Metaheuristics**.» communication nationale dans le cadre de la première Journée Nationale sur les Applications des Métaheuristiques (JNAM'07), le 29 mai 2007 à la Faculté d'Electronique et d'Informatique de l'U.S.T.H.B Alger.

[PLU03] J.P.W. Pluim and J.M. Fitzpatrick "**image registration**" Medical Imaging, IEEE Transactions on Volume 22, Issue 11, Nov. 2003 Page(s):1341 - 1343

[REI98] E. Reiffel, W. Polak « **An introduction to quantum computing for non-physicists**» august 16, 1998.

[ROB02] Marc ROBIN . "**Télédetection des Satellites aux SIG**" 2éme Edition, NATHAN Université 2002

[ROU98] J.-M. Rouet, J.-J. Jacq and C. Roux « **3D elastic multimodality image registration through a genetic algorithm** «Engineering Volume 2, 29 Oct.-1 Nov. 1998 Page(s):663 - 666 vol.2

[SAL01] M. SALOMON« **Étude de la parallélisation de méthodes heuristiques d'optimisation combinatoire application au recalage d'images médicales** » thèse, université Louis Pasteur 11 décembre 2001.

[SOO96] Soo-Young Lee; Kyung Geun Lee; "**Synchronous and asynchronous parallel simulated annealing with multiple Markov chains** »Parallel and Distributed Systems, IEEE Transactions on Volume 7, Issue 10, Oct. 1996 Page(s):993 - 1008

[SOU 04] Souquet A. et Radet Francois-Gérard "**Algorithmes génétiques** " 2004.

[SUN04] Youfa Sun and Feiqi Deng « **Chaotic parallel genetic algorithm with feedback mechanism and its application in complex constrained problem**" Cybernetics and Intelligent Systems, 2004 IEEE Conference on Volume 1, 1-3 Dec. 2004 Page(s):596 - 601 vol.1

[TAL04] Talbi, H.; Draa, A.; Batouche, M.; "**A new quantum-inspired genetic algorithm for solving the travelling salesman problem**" Industrial Technology, 2004. IEEE ICIT '04. 2004 IEEE International Conference on Volume 3, 8-10 Dec. 2004 Page(s):1192 - 1197 Vol. 3

[VAL01] Thomas Vallée et Murat Yildizoglu « **présentation des algorithmes génétiques et de leurs applications en économie** » V. 1.2. , 07 septembre 2001. URLbeagle.u-bordeaux4.fr/yildi/files/agpresf.pdf

[WAH00] Benjamin W. Wah and Yi Xin Chen « **optimal anytime constrated simulated annealing for constrained golobal optimization*** » department of Electrical and cumputer engineering and coordinated since laboratory, University of Illinois, urbana-chapaign, 1308 West main street Urbana, USA, URL http://www.manip.crhc.uiuc.edu

[YVE01] Yves Coueque - Julien Ohler - Sabrina Tollari « **Algorithmes génétiques pour résoudre le problème du commis voyageur**» avril 2001 URL http://sis.univ-tln.fr/~tollari/TER/AlgoGen1

[YOU98] Youcef CHIBANI « **Implémentation d'un processus de correction géométrique d'images satellitaires. Rectification carte image** » Magister en électronique des images.

[ZIT03] Zitova B. & Flusser J., "Image registration methods: a survey", Image and Vision Computing 21 (2003) 977–1000

[ZIV03] B. Zivota and J. Flisser "**image registration methods: a survey**" image and vision computing, vol. 21, pages: 977-1000, June 2003.