République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Larbi Tébessi - Tébessa
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département : Mathématiques et Informatique

Mémoire de fin d'étude
Pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Systèmes et Multimédias

Thème

Optimisation de la compression fractale par l'hybridation des métaheuristiques

Présentée Par :
MARZOUG Samir

Devant le jury :

Mr.DJEDDI Chawki	*MCA*	*Univérsité de Larbi Tébessi*	*Président*
Mr.SOUAHI M.Salah	*MAA*	*Univérsité de Larbi Tébessi*	*Examinateur*
Mr.MENASSEL Rafik	*MCB*	*Univérsité de Larbi Tébessi*	*Encadreur*

Date de soutenance : 22/06 /2019

Résumé

Un algorithme de compression d'images fractales basé sur l'optimisation d'essaims de particules hybrides avec algorithme génétique (PSO-GA) est proposé pour réduire l'espace de recherche. Adoptez l'optimisation des essaims de particules hybrides avec l'algorithme génétique (PSO-GA) pour explorer les optima globaux si les optima locaux ne sont pas satisfaits. Les résultats de l'expérience montrent que l'algorithme converge rapidement. Sur la base d'une bonne qualité de l'image reconstruite, l'algorithme a permis de gagner du temps de codage et d'obtenir un taux de compression élevé.

Les mots clés : Compression d'image fractale; essaim de particules optimisation; algorithme génétique; PSO-GA ;

Abstract

A fractal image compression algorithm based on hybrid particle swarm optimization with genetic algorithm (PSO-GA), is proposed to reduce the searching space. Adopt hybrid particle swarm optimization with genetic algorithm (PSO-GA), to explore the global optima if the local optima are not satisfied. Experiment results show that the algorithm convergent rapidly. At the premise of good quality of the reconstructed image, the algorithm saved the encoding time and obtained high compression ratio.

Keywords: Fractal image compression; particle swarm optimization ;genetic algorithm; PSO-GA;

ملخص

يُقترح خوارزمية لضغط الصور النمطي هندسيًا تعتمد على تحسين سرب الجسيمات الهجينة لتقليل مساحة البحث .استخدم تحسين سرب الجسيمات (PSO-GA) باستخدام الخوارزمية الجينية لاستكشاف أوبتيما العالمية إذا لم تكن راضية ، (PSO-GA) الهجينة باستخدام الخوارزمية الجينية أوبتيما المحلية .تظهر نتائج التجربة أن الخوارزمية تتقارب بسرعة .في فرضية الجودة الجيدة للصورة .التي أعيد بناؤها ، وفرت الخوارزمية وقت الترميز وحصلت على نسبة ضغط عالية

الكلمات المفتاحية:الخوارزمية الجينية ؛ سرب الجسيمات الهجينة ؛ ضغط صورة كسورية .
PSO-GA

Remerciments

Je tiens tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui ma a donné la force et la patience d'accomplir ce Modeste travail

J'exprime mes profonds remerciements à mon directeur de thèse, professeur R.MNASEL, pour toutes les informations qu'il m'a apporté, pour les conseils qu'il m'a donné, pour son suivi, sa disponibilité , sa sympathie ,sa patience et son intérêt porté sur le travail que j'ai réalisé.

Je tiens aussi à exprimer mon profonds remerciements au Membres de jury à leurs efforts et leur soin apporté à mon travail

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté à me rencontrer et répondre à mes questions durant mes recherches.

Je remercie très spécialement mes très chers parents,qui ont toujours été là pour moi, « Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous m'avez donné un magnifique modèle de labeur et de persévérance »

Je remercie également mes sœurs Maleke et Yossra et Hannan pour leur encouragement. Je tiens à remercier mes petits cousins ousama et mon frer Hamza.

Enfin, je remercie tous mes Ami(e)s que j'aime tant, Pour leur sincère amitié et confiance, et à qui je dois ma reconnaissance et mon attachement.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Dédicace

je dédie ce travail

A ma chère mère

A mon cher père

Qui n'ont jamais cessé , de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs .

A mes chères soeurs

A mes amis

Et ceux qui ont partagé avec moi tous les moments d'émotion lors de la réalisation de ce travail , il m'ont chaleureusement supporté et encouragé tout au long de mon parcours

A ma famille mes proches

Et à ceux qui m'aide et me supporté dans les moments difficiles.

A tous mes amis

qui m'ont toujours encouragé , et a qui je souhaite plus de succès.

A tous ceux que j'aime et ce qui m'aime.

Table de matière

Introduction générale……..01

Chapitre 01 :Compression d'image & Optimisation Etat de l'art

1.1 Introduction ……03

1.2 Notions de base ……....03

1.2.1 Définition d'une image ……03

1.2.2 Les différents types d'images …….... 04

1.2.3 Définition de compression ……..04

1.2.4 Besoins en compression ……..……05

1.3 Codage et décodage ……05

1.4 Les méthodes de compression ……06

1.4.1 La compression sans pertes ……....06

1.4.1.1 Modélisation des données……....06

1.4.1.2 Définition du codage entropique……..... 07

1.4.1.3 Codage de Shannon-Fano……....08

1.4.1.4 Le codage par plages (RLE) ……...……09

1.4.1.5 Codage de Hoffman ……..…. 09

1.4.1.6 Codage à base de dictionnaires ……..……10

1.4.1.7 Algorithme LZW ……..…….. 11

1.4.1.8 Le codage des images par plans de bits…….. 11

1.4.2 La compression avec pertes ……..…… 12

1.4.2.3 Quantification ……..… 12

1.4.2.3.1 Quantification scalaire …… 13

1.4.2.3.2 Quantification vectorielle ……....14

1.4.2.4 Codage par transformée …… 15

1.4.2.4.1 Transformation de Fourier Discrète ……..16

1.4.2.4.2 Transformation en Cosinus Discrète ……16
1.4.2.5 Le codage en sous-bandes ……17
1.4.2.6 La compression par ondelettes ……..18
1.4.2.7 Les normes de compression des images ……..….21
1.4.2.7.1 La norme de compression JPEG ……...……21
1.4.2.7.2 La norme de compression JPEG2000 ……..……..22
1.4.2.9 La méthode de compression fractale ……23
1.4.2.9.1 Les avantages et les inconvénients de compression fractale ……....26
1.4.2.9.2 Etude comparative des differentes methodes de compression principale ……..27
1.5 Métaheuristique……27
1.6 Méthodes métaheuristiques …… 29
1.6.1 Recuit simulé ……..29
1.6.2 Recherche Tabu ……30
1.6.3 Stratégie d'évolution ……30
1.6.4 Algorithmes génétiques ……31
1.6.5 Evolution différentielle ……..32
1.6.6 Algorithmes immunitaires ……33
1.6.7 Optimisation des essaims de particules ……34
1.6.8 Optimisation de la colonie de fourmis ……..35
1.6.9 Optimisation des colonies d'abeilles à miel ……..36
1.7 Hybridation de métaheuristiques ……..……...36
1.7.1 Les algorithmes Mémétiques ……....37
1.8 Conclusion ……..…38

Chapitre 02:Approche proposée

2.1 Introduction …… ……..39
2.2 Travaux connexes ……..39
2.2.1 Hybrid Genetic-Simulated Annealing Approach for Fractal Image Compression .39
2.2.2 Optimization of fractal image compression based on genetic algorithms ……40
2.2.3 Fractal image compression based on spatial correlation and hybrid genetic algorithm……40
2.2.4 Fractal image compression using upper bound on scaling parameter……40

2.2.5 Crowding Optimization Method to Improve Fractal Image Compressions Based Iterated Function Systems……41
2.2.6 Hybrid Fractal Image Compression Based on Graph Theory and Equilateral Triangle Segmentation……41
2.2.7 Cuckoo inspired fast search algorithm for fractal image encoding…… 41
2.2.8 A fractal image encoding method based on statistical loss used in agricultural image compression……42
2.2.9 Fractal image coding algorithm using particle swarm optimization and hybrid quadtree partition scheme……42
2.2.10 Fractal image compression based on spatial correlation and hybrid particle swarm optimization with genetic algorithm……43
2.2.11 An edge property-based neighborhood region search strategy for fractal image compression……43
2.2.12 Introducing BAT inspired algorithm to improve fractal image compression…..44
2.2.13 An improved fractal image compression using wolf pack algorithm……44
2.3 Approche proposée ……..44
2.3.1 Algorithme PSO-GA ……44
2.3.2 Algorithme Fractal ……....48
2.3.2.1 L'algorithme de Compression ……...48
2.3.2.2 L'algorithme de décompression ……..…..49
2.3.3 Méthode Proposée …… 49
2.4 Conclusion ……50
Chapitre 03 :Application d'une Hybridation(PSO-GA) à la compression fractale d'images
3.1. Introduction ……..51
3.2 Rapport et taux de compression ……..51
3.3 Mesure de la qualité visuelle de l'image reconstruite ……52
3.4 L'environnement de travail ……...…..52
3.4.1 Le système d'exploitation ……..…52
3.4.2 Ressources matérielles ……...53
3.4.3 Langage de programmation ……..53

3.4.4 Matlab et ses bibliothèques ……...53
3.4.5 Description de la fenêtre de commande de Matlab ……..54
3.5 Formulation du problème par PSO-GA ……..…55
3.6.1 Application De Hybride (PSO-GA) Au compression fractale D'images ……..56
3.6.2 Choix des paramètres ……57
3.6.2 Résultats et discussion ……...…58
Conclusion générale et perspective ……61

Liste des figures

Figure 1. 1 Système général de stockage d'images ……..05

Figure 1. 2 Le flux de base du codage de compression d'image……..06

Figure 1. 3 Codage de SHANNON -FANO……08

Figure 1. 4 Un exemple de codage par plage RLE……09

Figure 1. 5 Algorithme de Huffman……...10

Figure 1. 6 Principe de la quantification……12

Figure 1. 7 Quantification scalaire uniforme en escalier…….……13

Figure 1. 8- Principe du quantificateur vectoriel……..…14

Figure 1. 9 Schéma de principe de la compression / décompression par transformation.16

Figure 1. 10 Décomposition d'une image en sous-bandes……18

Figure 1. 11 Transformation en colonnes et en lignes……..19

Figure 1. 12 Transformation ondelette « pyramidal » (1-D (a),2-D (b),3-D(c))…….19

Figure 1. 13 Le principe de Compression / Décompression par ondelettes……20

Figure 1. 14 Schéma de principe de La compression JPEG……21

Figure 1. 15 Schéma typique d'un codeur JPEG 2000…….23

Figure 1. 16 Illustre le point fixe d'une transformation affine contractive par la répétition de l'application W…….24

Figure 1. 17 La fougère de Barnsley…….25

Figure 1. 18 Schéma Général d'un Codeur et Décodeur Fractale……..26

Figure 1. 19 Comparaison des méthodes en fonction (d'erreur RMSE, et taux de compression)…….27

Figure 1.20 Principe de métaheuristique……...29

Figure 1.21 Principe générale des méthodes hybrides……..37

Figure 2.1 Algorithme PSO-GA……47

Figure 2.2 Algorithme Fractale PSO-GA……..49

Figure 3.1 la fenêtre de commande de Matlab……..54

Liste des tableaux

Tableau 1. 1 Synthèse des différents algorithmes…….27

Tableau 1. 2-Pseudocode pour l'algorithme génétique ……...32

Tableau 1.3 -Pseudocode pour l'optimisation de l'essaim de particules…….35

Tableau 3.1 Les paramètres d'hybride (PSO-GA) ……..56

Tableau 3.2 Les paramètres de l'algorithme PSO……56

Tableau 3.3 Résultat de lina ……...56

Tableau 3.4 Résultat de barbara……....57

Tableau 3.4 Résultat de cameraman……..57

Tableau 3.5 Résultat de résolution linna……...58

Tableau 3.6 Comparaison la méthode proposer avec d'autre méthode ……...60

Introduction générale

Au cours des dernières années, le développement et la demande de produits multimédias se sont développés de plus en plus rapidement, contribuant à une bande passante insuffisante du réseau et au stockage d'un dispositif de mémoire.

Par conséquent, la théorie de la compression des données devient de plus en plus importante pour réduire la redondance des données afin d'économiser davantage d'espace matériel et de bande passante de transmission. En informatique et en théorie de l'information, la compression de données ou le codage est le processus de codage de l'information utilisant moins de bits ou d'autres unités portant de l'information qu'une représentation non codée.

La compression est utile car elle permet de réduire la consommation de ressources coûteuses telles que l'espace disque ou la bande passante de transmission. Nous introduisons brièvement la théorie fondamentale de la compression d'image.

D'autre part, les "métaheuristiques" sont des procédures conçues pour résoudre des problèmes d'optimisation dits difficiles. Ce sont en général des problèmes aux données incomplètes, incertaines, bruitées ou confrontés à une capacité de calcul limitée. Les métaheuristiques ont Connu le succès dans une large variété de domaines. Cela découle du fait qu'elles peuvent être appliquées à tout problème pouvant être exprimé sous la forme d'un problème d'optimisation de critère(s). Ces méthodes sont, pour la plupart, inspirées de la physique (recuit simulé), de la biologie (algorithmes évolutionnaires) ou de l'éthologie (essaims particulaires, colonies de fourmis).

Notre travail vient dans le cadre de combinaison des deux axes de recherche précédemment cités, et ce par la proposition d'une nouvelle méthode de compression basée sur une hybridation de deux métaheuristiques d'optimisation.

Ce mémoire est organisé comme suit :

Après une introduction qui va nous permettre de se positionner dans le contexte d'étude.

Introduction générale

Le chapitre 1, est consacré à la compression d'image et les algorithme méta- -heuristique. D'abord, les principes de base de l'imagerie numérique sont présentés. Ensuite, les diffèrent méthodes des compression perte est sans pertes et les avantages et les inconvénients de compression fractale.

Le chapitre 2, présente les différents travaux similaires dans le cadre de l'optimisation de la compression fractale par l'utilisation des métaheuristiques d'optimisation, suivi de l'explication de la méthode proposée qui se base sur l'algorithme hybride (PSO-GA).

Le chapitre 3 se consacrera à la description de l'application, ainsi que quelques résultats obtenus accentués par des discussions et des comparaisons possibles.

Nous terminerons ce mémoire par une conclusion et quelques futures perspectives.

Chapitre 01 : Compression d'image & Optimisation Etat de l'art

Chapitre 01 : Compression d'image & Optimisation Etat de l'art

1.1 Introduction

Depuis plus de 15 ans, la compression des données est pertinente en raison du volume croissant d'informations personnelles et professionnelles (documents, vidéos, son, images) que nous utilisons tous les jours. Les logiciels de compression de données sont utilisés en permanence pour stocker ou transmettre des informations avec des méthodes visant à réduire les informations redondantes dans le contenu des fichiers de données et, partant, à minimiser leur espace physique.

Les images numériques contiennent généralement des quantités très importantes de redondance spatiale et spectrale. La redondance spatiale est due à la corrélation entre les valeurs de pixels voisins et la redondance spectrale est due à la corrélation entre différents plans de couleur. Les techniques de compression d'image (codage) réduisent le nombre de bits requis pour représenter une image en tirant parti de ces redondances. Un processus inverse appelé décompression (décodage) est appliqué aux données compressées pour obtenir l'image reconstruite.

L'objectif de la compression est de réduire autant que possible le nombre de bits, tout en maintenant la résolution et la qualité visuelle de l'image reconstituée aussi proches que possible de l'image d'origine. Cet article donne un aperçu des principales techniques de compression d'image. Les étapes de décodage pour la plupart des schémas de codage sont assez intuitives et sont généralement l'inverse des étapes de codage.

Le présent chapitre est organisé en deux sections, la première section est consacrée à la description la compression sans pertes et les différentes méthodes utilisées dans le même type de compression. Dans la deuxième section nous présenterons un état de l'art des méthodes avec pertes les plus connues tout en mettant l'accent sur la compression fractale, ses avantages et ses inconvénients.

1.2 Notions de base

1.2.1 Définition d'une image

Une image est la reproduction d'une scène obtenue à l'aide de systèmes de production d'images (appareils photographique, caméra, radiographies, scanner, sonar, ...). Sa forme peut être analogique (ex : photographie, vidéo...) ou digitale (images numérisées suivant divers

formats, images compressées ou non…) ou obtenues par des capteurs fournissant des images numérisées) et dans ce cas un traitement par ordinateur est possible [1].

Sonka et al [2] disent que l'image "peut être modelée par une fonction continue de deux ou trois variables ; dans le cas simple les arguments sont les coordonnées (x, y) dans le plan, alors que si les images changent dans le temps une troisième variable ***t*** peut être ajoute. Les valeurs de fonction d'image correspondent à la luminosité aux points de l'image ; ceci nous permet d'éviter la description du processus très compliqué de la formation d'image."

1.2.2 Les différents types d'images :

On distingue différents types d'images [3]:

1- L'image en niveaux de gris : Image 8 bits (entier de 0-255) ;

2- Mode monochrome : Image 16 bits (entier de 0-65535) ;

3- L'image en couleurs : Image 32-bits (réel).

1.2.3 Définition de compression :

La compression est le processus de réduction de la taille d'un fichier de données à l'aide d'algorithmes permettant de restructurer la manière dont les données sont organisées dans le fichier. La compression est utilisée généralement pour faciliter le stockage et le transfert de gros fichiers. Le fichier résultant peut conserver toutes les données ou des données, y compris des informations visuelles, peuvent être perdues. Les algorithmes de compression qui conservent toutes les données d'origine sont dits « sans perte » et ceux dans lesquels des données sont perdues sont appelés « avec perte ». En réglant l'appareil photo ou le logiciel sur une compression minimale (ou sur le moins d'images que vous pouvez stocker), vous réduirez considérablement la quantité de données perdues. La décision d'utiliser une compression avec ou sans perte sera dictée par l'utilisation prévue de l'image. [4], [5]

La Figure 1.1 [6]montre le schéma fonctionnel du système général de stockage d'images. L'objectif principal d'un tel système est de réduire autant que possible la quantité de stockage, et l'image décodée affichée sur le moniteur peut être autant que possible similaire à l'image d'origine. L'essence de chaque bloc sera présentée dans les sections suivantes.

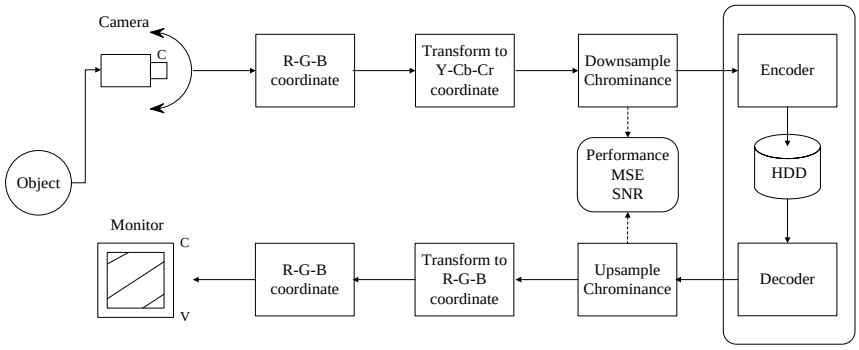


Figure 1. 1 Système général de stockage d'images

1.2.4 Besoins en compression :

- Une image couleur représentée dans l'espace Rouge-Vert-Bleu, de taille 512*512 pixels, dont chacune des composantes est codée sur 8 bits par pixel représente 786 Kilo-octets ;
- Un film négatif 24*36 mm numérise à 12 µm par point retourne une image de taille 3000*2000 pixels par couleur,8bpp,3 couleurs, ce qui représente 18 Méga-octets ;
- Une image LANDSAT ; 6000*6000 pixels par bande spectrale, 8bpp, 6 bandes, représente 216 Méga-octets ;
- La transmission d'une séquence vidéo 512*512 ,8bpp, 3couleurs sur une ligne téléphonique avec un modem a 9600 bauds nécessite 11 minutes par image ;
- La transmission d'une séquence d'images couleur au format QCIF échantillonnée à 30 Hertz représente un débit de 9.12 Méga bits par seconde, et de 36.40 Méga bits par seconde au format CIF (Common Intermédiate Format).

1.3 Codage et décodage :

Le codage par compression d'image consiste à stocker l'image dans un flux de bits aussi compact que possible et à afficher l'image décodée sur le moniteur aussi précisément que possible. Considérons maintenant un codeur et un décodeur, comme illustré à la Fig. 1.2. Lorsque l'encodeur reçoit le fichier image d'origine, celui-ci est converti en une série de données binaires, appelée flux de bits. Le décodeur reçoit alors le train de bits codé et le décode pour former l'image décodée. Si la quantité totale de données du train de bits est inférieure à la

quantité totale de données de l'image d'origine, on parle alors de compression d'image. Le flux de compression complet est comme indiqué à la Figure 1.2[6].

Figure 1. 2 Le flux de base du codage de compression d'image

1.4 Les méthodes de compression

Les méthodes de compression visent à enlever la redondance présente dans l'image de manière à diminuer le nombre débits nécessaires à sa représentation Plusieurs types de redondance en termes de corrélation peuvent être considères :

- La redondance spatiale entre pixels ou blocs voisins dans l'image
- La redondance spectrale entre plans de couleur ou bandes spectrales
- La redondance temporelle entre images successives dans une séquence vidéo

Les méthodes de compression peuvent se regrouper en deux classes

- Les méthodes sans perte d'information (réversibles) : le taux de compression est limité par l'entropie de l'image
- Les méthodes avec perte d'information(irréversibles) : le taux de compression est sensiblement supérieur à l'entropie de l'image.

1.4.1 La compression sans pertes :

La compression sans perte signifie que lorsque les données sont décompressées, le résultat est une correspondance parfaite bit par bit avec l'original. Le nom « lossless » en anglais signifie "aucune donnée n'est perdue", les données sont seulement sauvegardées de manière plus efficace dans leur état compressé, mais rien n'est supprimé.

1.4.1.1 Modélisation des données

Les méthodes de codage classiques élaborent une table de probabilités statique avant de construire le code. La table est calculée lors du codage puis transmise au décodeur. Elle

peut aussi être calculée une fois pour toutes, servant ainsi au codage de plusieurs réalisations d'une source (images). Mais l'utilisation d'un tel modèle statique est dangereuse car le flot d'entrée peut ne pas correspondre aux statistiques précédemment calculées. Le taux de compression est diminué. Une meilleure compression peut être atteinte en augmentant l'ordre de la modélisation (extension d'ordre plus élève de la source). Le gain en compression réalise est dans ce cas annulé par la taille exponentiellement croissante de la table de probabilités. Une image contenante 256niveaux de gris, vue comme une réalisation d'une source simple (ordre 0) nécessite la construction d'une table de 256probabilités. Une table d'ordre 1 contient quant à elle 65536 probabilités. Pour ces raisons, on utilise aujourd'hui des modèles adaptatifs. Les statistiques sont continuellement modifiées au cours de la lecture des symboles à coder. L'avantage de ces méthodes par rapport à celles basées sur des modèles statiques est leur capacité d'adaptation aux conditions locales.

1.4.1.2 Définition du codage entropique

Considérons une image dont chaque pixel est quanti sur B bits. L'image (monochrome) possède 2^B niveaux de gris différents notes s_i ($i = 1 : : : 2^B$). Si les niveaux de gris sont indépendants et équiprobables (probabilité du pixel $i = p_i = 2^{-B}$)), l'entropie H de l'image est égale à $\log_2 2^B = B$ bits. Chaque pixel porte la même quantité d'information, l'image est incompressible". Si les niveaux de gris ne sont pas équiprobables (densité de probabilité non uniforme), l'entropie H est inferieure a B. Le but du codage entropique est de coder les pixels à l'aide de mots code de Longueurs variables, égalés a $-\log_2 p_i$ bits, de manière à ce que le nombre moyen de bits par pixel soit égal à l'entropie $H = -\sum_{i-1}^{2B} p_i \log_2(p_i)\ bits\ \left(H \leq B\right).$

(1.1)

Le codage est dans le cas général sous-optimal car la quantité $(-\log_2 p_i)$ est rarement entière. La limite H peut être approchée en codant des extensions de la source (en regroupant les pixels).

1.4.1.3 Codage de Shannon-Fano

C. Shannon du laboratoire Bells et R.M. Fano du MIT ont développé à peu près en même temps une méthode de codage basée sur la simple connaissance de la probabilité d'occurrence de chaque symbole dans un message.

Algorithme de construction de la table de codes irréductibles

1. Les probabilités d'occurrence de chaque message sont placées dans une liste dans un ordre décroissant. La liste constitue la racine d'un arbre qui, pour l'instant, est une feuille.
2. Couper la liste en deux groupes de symboles S_0 et S_1, dont les probabilités totales sont aussi voisines que possible ($\simeq \frac{1}{2}$).
3. Le groupe S_0 est code par un "0", le groupe $\bar{S}_1$ par un "1".
4. Si un groupe S_i n'a qu'un seul élément, il est appelé "feuille terminale" et est inchangé. Sinon, la procédure reprend à l'étape 2 sur le groupe S_i.

La procédure de codage construit un arbre dont les suites de bits 1 ou 0 partant de la racine vers chacune des feuilles constituent les mots code du code (Figure1.3).

Octet	Fréquence processus	Code
C	11 — 11	0
B	7 — 7	10
E	5 — 0 — 5	110
D	4 — 1 — 18 — 0 — 6 — 0 — 4	1110
A	2 — 1 — 11 — 1 — 1 — 2	1111

Figure 1. 3 Codage de SHANNON -FANO

1.4.1.4 Le codage par plages (RLE) :

Appelé en anglais **« run_length encoding »**, Une nouvelle approche de la compression des images en noir et blanc est décrite. Elle permet de compresser les huit documents de test sans perte de 20 à 30% de mieux que les meilleurs algorithmes de compression existants. Les aspects de codage et de modélisation sont traités séparément. La clé de ces améliorations est un code arithmétique binaire efficace. Le code est relativement simple à mettre en œuvre car il évite l'opération de multiplication inhérente à certains codes arithmétiques antérieurs(voir[7]). Le codage arithmétique permet la compression de séquences binaires dont les statistiques changent bit par bit. Les statistiques modèles sont étudiées à partir d'hypothèses stationnaires, adaptatives stationnaires et adaptatives non stationnaires, et cet algorithme utilise dans les texte volumétrique dans document[8].

Example :

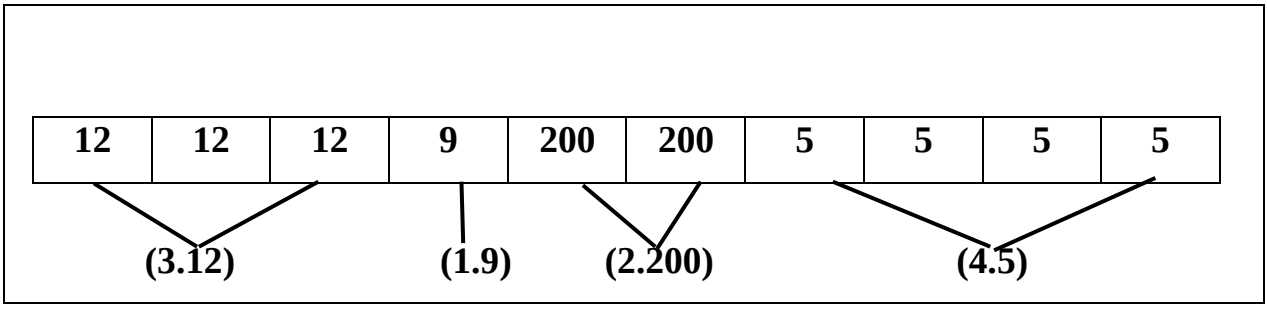


Figure 1. 4 Un exemple de codage par plage RLE

1.4.1.5 Codage de Hoffman

Le principe consiste à utiliser un nombre de bits inférieur pour coder les données qui se produisent plus fréquemment. Les codes sont stockés dans un livre de codes pouvant être construit pour chaque image ou un ensemble d'images. Dans tous les cas, le livre de codes et les données codées doivent être transmis pour permettre le décodage(voir[9]). La clé est que l'encodeur et le décodeur utilisent exactement les mêmes routines d'initialisation et de mise à jour du modèle. Le modèle de mise à jour fait deux choses :

(a) Incrémenter le nombre.

(b) mettre à jour l'arbre de Hoffman.

Au cours des mises à jour, l'arbre de Hoffman conservera sa propriété Le principe illustré par la figure 1.5.

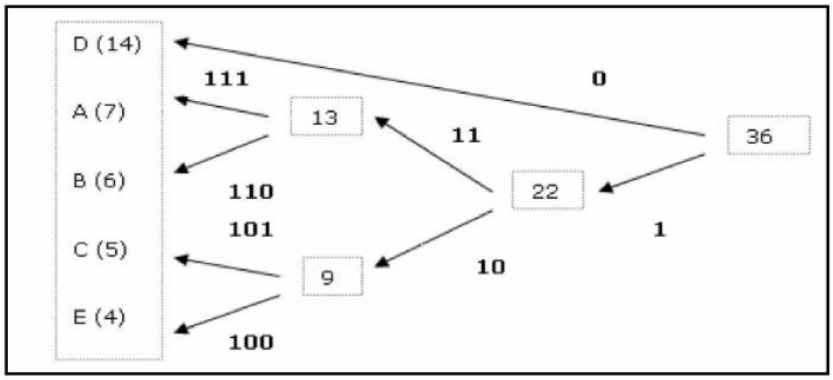


Figure 1. 5 Algorithme de Huffman

Le codage obtenu est donc : D (occurrence =14) : 0 ;

A (occurrence =7) : 111 ; B (occurrence =6) : 110 ;

C (occurrence =5) : 101 ; E (occurrence =4) : 100

1.4.1.6 Codage à base de dictionnaires

Les méthodes telles que celle de Human, Shannon-Fano sont basées sur des modelés statistiques plus ou moins complexes (ordres 0,1,2, ou modelés adaptatifs). Les méthodes de compression par dictionnaire ne traitent quant à elles pas directement une chaine de bits codant un symbole mais codent un index pointant sur la chaine dans un dictionnaire de symboles.

Les méthodes statiques utilisent un dictionnaire prédéfini pour coder les symboles d'un chier. Le dictionnaire est construit avant que la compression ne commence, en rassemblant un échantillon de symboles représentatifs. Par exemple, si on veut coder un chier texte écrit en langage C, le dictionnaire doit se concentrer sur des mots comme "Read", "while", "printf". Le dictionnaire est utile au codeur et au décodeur car ceux-ci respectivement génèrent et reçoivent un index pointant sur un mot dans le dictionnaire.

Les méthodes adaptatives : La compression commence sans dictionnaire, ou avec un dictionnaire minimum par défaut. Lors de l'évolution de l'algorithme, des nouvelles phrases, ou de nouveaux mots sont ajoutes, utilises plus tard pour coder d'autres mots.

1.4.1.7 Algorithme LZW :

LZW doit son nom à Abraham Lempel, Jakob Ziv et Terry Welch, les scientifiques qui ont développé cet algorithme de compression.

La compression LZW remplace les chaînes de caractères par des codes uniques. Il ne fait aucune analyse du texte entrant. Au lieu de cela, il ajoute simplement chaque nouvelle chaîne de caractères qu'il voit à une table de chaînes.

La compression se produit lorsqu'un seul code est généré à la place d'une chaîne de caractères.

La compression LZW fonctionne mieux pour les fichiers contenant beaucoup de données répétitives. C'est souvent le cas avec du texte et des images monochromes. Les fichiers compressés mais ne contenant aucune information répétitive peuvent même grossir!(voir[10]) .

L'algorithme suivant montre son mécanisme

Le LZW ou LZ77
W ◄— NULL
Tant que (valeur # eof)
  Lire la valeur k
Si w_k est dans le dictionnaire w ◄— w_k
Sinon Ecrire le code de W
Ajouter w_k au dictionnaire
W◄— K
Fin tant que

1.4.1.8 Le codage des images par plans de bits

Une image codée sur 2^n niveaux de gris (n bits par pixel) peut être considérée comme une superposition de n plans, chacun de hauteur 1 bit, en isolant à chaque fois un bit de même

poids pour chaque pixel. Chaque plan de bits (image binaire) peut être code séparément en utilisant la méthode RLE. Le code de Gray est dans ce cas utilise pour augmenter la cohérence au sein des différents plans de bits.

Le codage réversible par plans de bits permet des taux de compression compris entre 1.5 et 2, mais présente l'inconvénient d'^être sensible vis à vis des erreurs de transmission. Les plans composes des bits de poids fort contiennent la majeure partie de l'information visuelle de l'image.

1.4.2 La compression avec pertes :

La compression avec perte signifie que certaines données sont perdues lors de la décompression. La compression avec pertes repose sur l'hypothèse que les fichiers de données actuelles enregistrent plus d'informations que l'être humain ne peut « en percevoir ». Ainsi, les données non pertinentes peuvent être supprimées.

1.4.2.3 Quantification :

En général, la quantification apparaît en deuxième lieu dans un processus de compression, pour réduire la quantité d'information, de manière souvent irréversible [11]. Ainsi, une quantification est utilisée pour simplifier la représentation, tout en préservant l'information la plus pertinente. On remplace ainsi les valeurs initiales par un ensemble fini d'éléments qui donneront des résultats acceptables lors de la phase de décompression. Ces éléments peuvent être scalaires lorsque, on remplace l'information en chaque pixel par des nombres entiers ; ou des vecteurs lorsque l'on considère des groupes de pixels représentant des configurations typiques.

Dans les deux cas, le principe est décrit par la Figure 1.6.[12] Un élément quelconque (scalaire ou vecteur) est remplacé par l'élément du dictionnaire le plus approprié.

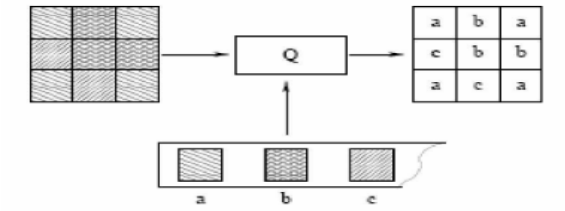


Figure 1. 6 Principe de la quantification

1.4.2.3.1 Quantification scalaire :

La quantification scalaire [13], consiste à réduire le nombre de valeurs que peut prendre une grandeur scalaire. C.à.d. remplacer un nombre très grand de symboles par un nombre restreint de codes. C'est une opération irréversible très largement employée en compression (il est à noter que les méthodes de codage utilisant un quantificateur ne sont jamais réversibles parce que l'étape de quantification introduit inévitablement une distorsion).

Un quantificateur scalaire est un opérateur qui associe à une variable continue u une variable discrète 'u pouvant prendre un nombre plus faible, et fini de valeurs ; il est généralement défini comme une fonction en escalier (Figure 1.7)[12], l'intervalle entre chaque valeur étant appelé intervalle de décision.

Pour un nombre de niveaux de quantification fixé L, on peut choisir les régions de décision {t k, k = 1.... L + 1} ainsi que les seuils de décision {r 1.....r} de façon à minimiser la L distorsion entre l'entrée et la sortie. Si u se trouve dans la région (t k, t k+1), u' aura pour valeur r.

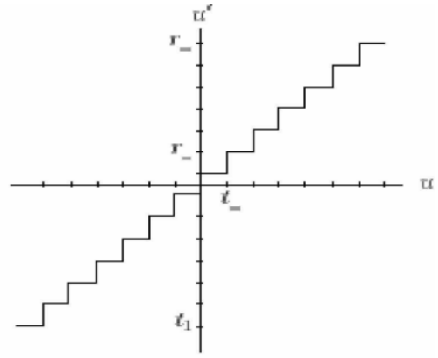


Figure 1. 7 Quantification scalaire uniforme en escalier

La quantification scalaire porte sur la grandeur physique associée à l'information de niveau de l'image. L'objectif de la quantification est de diviser la gamme dynamique ou dynamique de la grandeur physique en un nombre fini d'intervalles, et d'attribuer à toutes les valeurs du même intervalle une seule valeur, dite valeur quantifiée. Trois questions se posent alors :

- ✓ Trouver la gamme dynamique ;
- ✓ Choisir le nombre d'intervalles ;

- ✓ Répartir ces intervalles ;

Il existe plusieurs formes de quantifications scalaires dont la plus simple est la quantification scalaire uniforme. C'est la forme la plus couramment utilisée en compression d'images.

1.4.2.3.2 Quantification vectorielle :

Les techniques de compression d'images exploitent généralement la redondance statistique présente dans l'image. Les différentes méthodes vues jusqu'ici effectuent une quantification scalaire sur des échantillons (pixels individuels de l'image). La compression par transformée (DCT …) quantifie les échantillons issus de blocs transformes d'images. Le codage prédictif quantifie une erreur (différence) entre l'échantillon courant et sa prédiction. Mais ces valeurs ne sont jamais totalement décorrélées, ou indépendantes. Shannon a montré qu'il était toujours possible d'améliorer la compression de données en codant des vecteurs plutôt que des scalaires, cela étant vrai même si la source est sans mémoire (simple). La quantification vectorielle, développée par Gersho et Gray (1980) fait aujourd'hui l'objet de nombreuses publications dans le domaine de la compression des images numériques [13].

Principe de la quantification vectorielle

Un quantificateur peut être vu comme une application Q associant à chaque vecteur d'entrée

$X_i = (x_i\ j = 1 : : : k)$ un vecteur $Y_i = (y_i\ j = 1 \dots k) = Q(X_i)$

Choisi parmi un dictionnaire de taille nie, $C = (X_l, l = 1 \dots N_c)$. C peut être vu comme un catalogue de formes(Figure 1.8)[12] .

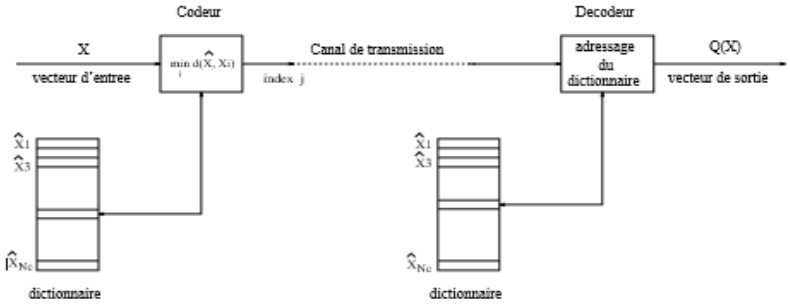


Figure 1. 8- Principe du quantificateur vectoriel.

1.4.2.4 Codage par transformée :

Dans ces méthodes, l'image de dimension NxN est subdivisée en sous images ou blocs de taille réduite (la quantité de calcul demandée pour effectuer la transformation sur l'image entière est très élevée). Chaque bloc subit une transformation mathématique orthogonale inversible linéaire du domaine spatial vers le domaine fréquentiel, indépendamment des autres blocs (transformée en un ensemble de coefficients plus ou moins indépendants). Les coefficients obtenus sont alors quantifiés et codés en vue de leur transmission ou de leur stockage. Pour retrouver l'intensité des pixels initiaux, on applique sur ces coefficients la transformation inverse.

L'objectif de ces transformations est double : il s'agit de

- ✓ Décorréler les données, c'est-à-dire d'obtenir des coefficients transformés moins corrélés que les pixels de l'image ;
- ✓ Concentrer l'énergie sur un nombre réduit de coefficient, les coefficients ayant une valeur plus importante aux basses fréquences qu'aux hautes fréquences.

Dans ce cas, on obtiendra une compression effective en codant finement les coefficients des basses fréquences, et grossièrement, voire en les supprimant, les coefficients hautes fréquences. Dans ce but, plusieurs transformations linéaires ont été proposées :

• Transformation de Karhunen-Loeve (TKL).

• Transformation de Fourrier discrète (TFD).

• Transformation de Hadamard (TH).

• Transformation de Haar (THA).

• Transformation en cosinus discrète (TCD).

- ✓ L'efficacité de ces transformations peut être mesurée par la prise en compte de trois facteurs : l'efficacité de décorrélation, la concentration de l'énergie, et l'existence d'algorithmes rapides pour calculer les transformations.
- ✓ Les méthodes de codage par transformation présentent des propriétés d'immunité au bruit de transmission bien supérieures aux méthodes précédentes. Une erreur affecte en

effet seulement la valeur d'un coefficient, et sera lissée au décodage lors du calcul de la transformation inverse. L'effet sera ainsi peu visible.

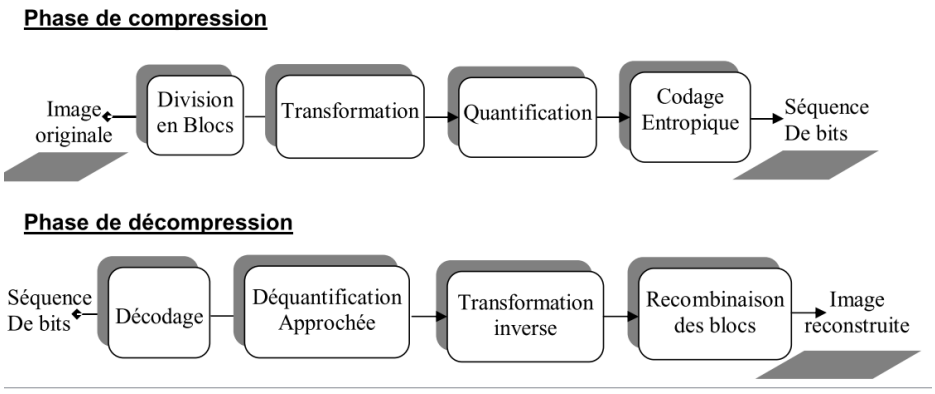


Figure 1. 9 Schéma de principe de la compression / décompression par transformation

1.4.2.4.1 Transformation de Fourier Discrète : DFT

Elle permet simplement de passer du domaine spatial au domaine fréquentiel. Cette transformée rend donc visible les composantes en fréquence de l'image

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) \cdot e^{-j\varpi . t} \cdot dt \quad \textbf{(1.2)}$$

Il est intéressant de noter que nous pouvons revenir au domaine spatial via la transformée de fourrier inverse. Une application brute de cette formule étant extrêmement longue. Une autre façon d'effectuer ce calcul permet de limiter considérablement la durée de cette transformation. C'est ce que l'on appelle la FFT (Fast Fourier Transform) [im27].

1.4.2.4.2 Transformation en Cosinus Discrète : DCT

Lorsque les images présentent une forte corrélation entre pixels voisins, les images de base de la DCT tendent à être égales à celles de la transformation de Karhunen Loeve (ces dernières dépendant de l'image à traiter) [14].

La DCT (Discret Cosinus Transform) est une transformée fort semblable à la FFT, travaillant sur un signal discret. Elle prend un ensemble de points d'un domaine spatial et les transforme en une représentation équivalente dans le domaine fréquentiel. La DCT transforme un signal d'amplitude (chaque valeur du signal représente l'amplitude d'un phénomène) discret bidimensionnel en une information bidimensionnelle de "fréquences".

Les équations qui suivent, donnent respectivement la transformée en cosinus discrète directe et inverse.

$$DCT(i,j) = \frac{1}{2\sqrt{2N}} C(i)c(j) \sum_{X=0}^{N-1} \sum_{Y=0}^{N-1} pixel(X,Y) \cos\left[\frac{(2X+1)i\pi}{2N}\right] \cos\left[\frac{(2Y+1)j\pi}{2N}\right]$$

$$pixel(X,Y) = \frac{1}{2\sqrt{2N}} \sum_{X=0}^{N-1} \sum_{Y=0}^{N-1} C(i)c(j)DCT(i,j) \cos\left[\frac{(2X+1)i\pi}{2N}\right] \cos\left[\frac{(2Y+1)j\pi}{2N}\right] \quad \textbf{(1.3)}$$

Ou

$C(i) = 1/\sqrt{2}$ si i =0 ; C (i)=1 si i $\neq$ 0

Pixel (X, Y) désigne la valeur du pixel de cordonnées (X, Y) et DCT (i, j) le coefficient repéré par la ligne i et la colonne j dans la matrice DCT.

La TCD est effectuée sur une matrice carrée NxN de valeurs de pixels en donnant une matrice carrée NxN de coefficients de fréquence. Le temps de calcul requis pour chaque élément dans la TCD dépend de la taille de la matrice.

Vu la difficulté d'appliquer la TCD sur la matrice entière, celle-ci est décomposée en blocs de taille 8x8 pixels (compression JPEG). Concrètement, et en terme simple, cette transformation va essayer de faire correspondre des blocs de 8x8 de l'image en une somme de fonction basique qui sont données dans la matrice 8x8 de la DCT (DCT matrix).

Différence entre la FFT et la DCT

- ✓ La DCT est actuellement une version simplifier de la FFT.
- ✓ Seule la partie réelle de la FFT est conservé
- ✓ Beaucoup plus simple en termes de coup de programmation
- ✓ La DCT est efficace dans la compression de multimédia (JPEG)
- ✓ DCT Beaucoup plus utilisée.

1.4.2.5 Le codage en sous-bandes

Le codage en sous-bandes a été introduit pour la parole par Crochiere et al. En 1976, puis appliqué à l'image sous la forme que nous connaissons aujourd'hui par Woods et O'Neil en 1986 [15]. Il consiste à décomposer le signal ou l'image en différentes bandes de fréquences spatiales et à coder chacune d'entre elles de manière indépendante ou non. Dans les schémas

classiques, la bande spectrale du signal original est d'abord divisée en deux parties à l'aide de deux filtres numériques (passe-bas et passe-haut).

La procédure est ensuite répétée dans chacune des sous-bandes fréquentielles. Le résultat constitue une structure arborescente, symétrique ou non, composée d'un nombre entier de sous-bandes. Le signal est reconstruit par recombinaison des sous-bandes à l'aide de filtres d'interpolation.

La figure 1.10[12] représente une étape de la décomposition. Le banc de filtre d'analyse qui est généralement un banc bi-dimensionnel suivis chacun par sous-échantillonneur. Ces filtres uni- dimensionnels et ces sous-échantillonnages, sont d'abord appliqués aux lignes de l'image, et ensuit aux colonnes des images obtenues. On se retrouve avec quatre images notées. L'image centre BF est une sous-image de basse résolution, sous-image horizontale HF, sous-image verticale HF et sous-image diagonale HF correspondent à quatre sous-bandes fréquentielles directionnelles.

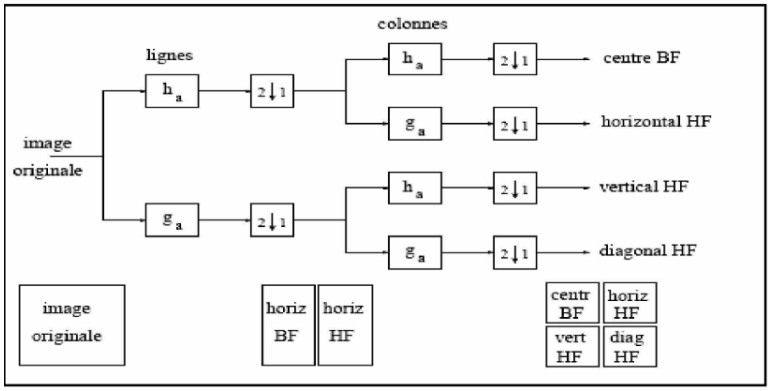


Figure 1. 10 Décomposition d'une image en sous-bandes

Ces travaux ont suscité un grand intérêt pour les techniques multi résolution qui se sont traduits par les travaux sur l'utilisation des ondelettes en analyse du signal (Rioul et Vetterli) dans les années 80, et de nombreuse application au codage de la parole, de la musique et de l'image fixe et animée. C'est un domaine encore très actif aujourd'hui.

1.4.2.6 La compression par ondelettes

Les ondelettes c'est d'abord une théorie mathématique récente d'analyse du signal, développée dans les années 80. On peut considérer qu'il s'agit d'une extension de l'analyse de Fourier [16]. La technologie de compression à base d'ondelettes offre une plus grande finesse

au niveau de l'analyse du signal, et permet de mieux s'adapter aux propriétés locales de l'image. La transformation par Ondelettes est une technique de compression d'image fixe très performante.

Quand on enregistre une image en utilisant les ondelettes, on divise sa résolution par deux et on code l'information perdue par des coefficients d'ondelettes. On commence donc à coder les détails les plus fins (les hautes fréquences). On recommence l'opération autant de fois que nécessaire jusqu'à ce que l'image se réduise à 1 pixel. A chaque étape l'image est « lissée » et les détails perdus sont codés en coefficients d'ondelettes. L'intérêt de la transformation par ondelettes par rapport aux autres méthodes de compression est que celle-ci ne considère pas l'image dans son ensemble pour la coder mais, la travaille par couche, cherchant à enregistrer les détails les plus importants (ceux qui se démarquent le plus du reste du signal) à chaque résolution [im26].

Algorithme de compression par ondelette ou waveletes :

Passons maintenant à l'algorithme pyramidal utilisé. La décomposition en coefficients d'ondelettes n'utilise pas une fonction de moyenne, mais s'appuie sur deux filtres. Un filtre passe bas (H) et un filtre passe haut (L). La combinaison de ces filtres permet d'obtenir quatre sous images HH, HL, LH et LL. Ces filtres sont nommés miroirs et quadratures (Figure 1.11)[12].

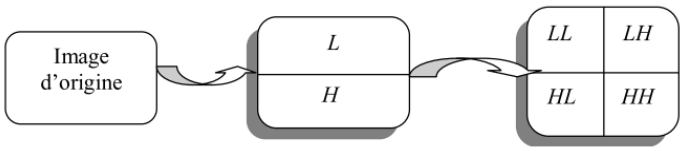


Figure 1. 11 Transformation en colonnes et en lignes

Chacune des quatre images obtenues par la transformation représente des informations bien distinctes.

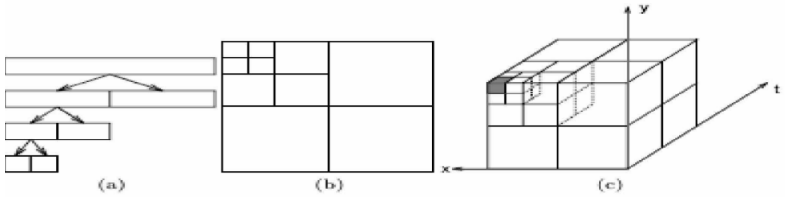


Figure 1. 12 Transformation ondelette « pyramidal » (1-D (a),2-D (b),3-D(c))

Les étapes de compression par ondelettes :

Tout d'abord, voici le principe de compression et décompression d'une image par ondelettes (Figure 1.13) [12]:

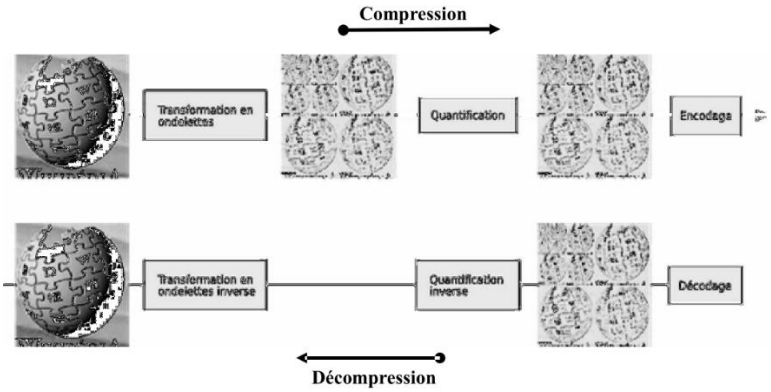


Figure 1. 13 Le principe de Compression / Décompression par ondelettes

La compression ondelettes :

Les étapes pour compresser une image sont

1. Transformations par ondelettes

2. Quantification : les valeurs des images de détails inférieures à un certain niveau sont éliminées, en fonction de l'efficacité recherchée. C'est cette étape qui introduit des pertes.

3. Codage des valeurs restantes. Les données restantes sont transmises à un encodeur entropie, c'est à dire à un algorithme de compression de données (LZW, HUFFMAN, …).

Décompression ondelettes :

La transformation inverse par ondelettes reconstruit une image originale. La construction de l'image à partir des sous-bandes restitue l'image en mode progressif. L'affichage de l'image peut s'effectuer en deux modes :

- ✓ Soit la taille de l'image augmente au fur et à mesure de la lecture du fichier compressé.
- ✓ Soit la résolution de l'image augmente au fur et à mesure de la lecture du fichier compressé.

1.4.2.7 Les normes de compression des images :

1.4.2.7.1 La norme de compression JPEG :

La norme JPEG (Joint Photographic Experts Group) est conçue par le groupe ISO (International Standards Organisation) et le groupe CEI (Commission Electronic International). Ce standard a vu le jour en 1992. Il est devenu le format le plus populaire pour le stockage de photographies numériques, car il offrait alors une meilleure compression que tous les formats bitmap connus à l'époque. Le standard avait pour objectif de concevoir une nouvelle norme de compression, avec les différentes contraintes suivantes [17]:

- ✓ La norme JPEG doit être très proche des nouvelles techniques de compression, en termes de taux de compression, qualité de restitution et de temps de calcul ;
- ✓ JPEG doit pouvoir compresser tout type d'images réelles (tailles différentes, images multi-composantes . . .) ;
- ✓ L'algorithme doit être implémentable sans trop de problèmes sur une grande gamme de CPUs et sur des cartes plus spécialisées ;
- ✓ Le codage doit pouvoir être séquentiel, progressif, sans perte et/ou hiérarchique,

Les techniques définies par la norme JPEG se divisent en deux classes : les méthodes de compression avec pertes qui sont basées sur la TCD suivie d'une quantification et d'un codeur entropique. La seconde classe, concerne les processus de codage sans pertes, cette classe de codeurs n'est pas basée sur la TCD mais sur le codage MICD suivi d'un codage entropique.

La figure 1.14 [12]explique clairement les différentes étapes de l'algorithme de compression JPEG non réversible.

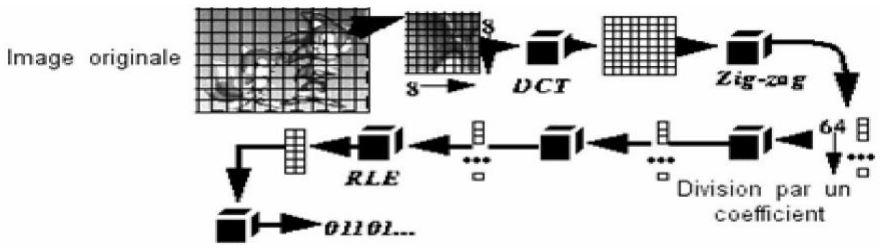


Figure 1. 14 Schéma de principe de La compression JPEG

La décompression JPEG

Comporte les mêmes étapes que la compression mais dans le sens inverse en commençant par la méthode de décodage statistique, puis la matrice obtenue est multipliée par la matrice de quantification que l'on reconstitue grâce au facteur de qualité et enfin on applique la DCT inverse (IDCT) pour retrouver une image plus ou moins dégradée par rapport à l'image initiale**.**

Algorithme JPEG réversible

La seconde classe, concerne les processus de codage sans pertes, cette classe de codeurs n'est pas basée sur la TCD mais sur le codage MICD suivi d'un codage entropique (huffman ou arithmétique). Le traitement se fait sur l'image entière, et non sur des blocs.

1.4.2.7.2 La norme de compression JPEG2000 :

La version définitive du standard a pris forme en décembre 2000. Ce nouveau standard a pour objectif d'offrir de nouvelles fonctionnalités permettant de répondre à une demande croissante, à savoir :

- Obtenir des performances de compression supérieures à son prédécesseur JPEG, notamment pour des débits très faibles.
- Permettre d'organiser le fichier compressé de plusieurs manières, notamment en fonction de la résolution désirée ou de la qualité de reconstruction.
- Avoir un mode de compression sans perte performant
- Fournir la possibilité de coder des parties d'une image avec une qualité supérieure à d'autres parties [15].
- En JPEG 2000 le même algorithme de codage s'applique à une grande variété de types d'images (couleurs, niveaux de gris, multi-composantes...).

La norme JPEG 2000 est une norme très récente basée sur la technologie des ondelettes. Le résultat pratique de ce choix devrait être une amélioration modeste mais réelle du taux de compression (de l'ordre de 20 %) et une qualité supérieure de restitution par rapport à ce que permet le JPEG actuel. Les coefficients d'ondelettes sont plus faciles à stocker que les blocs résultants de l'application de la DCT [im20]. Cette nouvelle norme est composée de plusieurs parties dont certaines sont publiées et d'autres non.

Les algorithmes utilisés :

Cet algorithme typique de codage JPEG 2000 se décompose essentiellement en 5 modules (Figure 1.15)[12]:

- Transformée couleur
- Transformée en ondelettes discrète
- Quantification
- Codage entropique
- Allocation de débit.

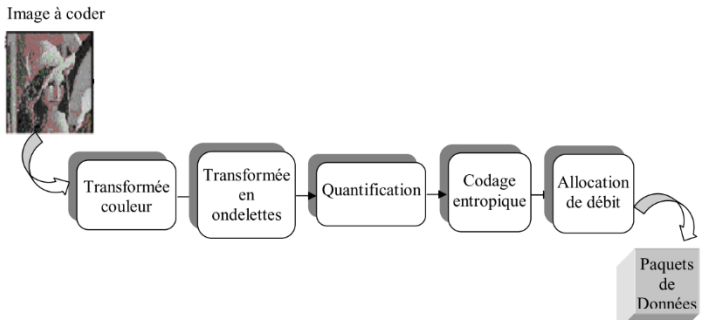


Figure 1. 15 Schéma typique d'un codeur JPEG 2000

Pour plus de détail sur le standard de compression JPEG 2000 voir l'article [18].

1.4.2.9 La méthode de compression fractale :

La compression fractale des images est une technique relativement récente puisqu'elle a été proposée pour la première fois par M. Barnsley en 1988 « fractals everywhere ». En effet L'idée consiste donc à changer la représentation de l'image : au lieu de la représenter à l'aide de pixels, on le fait à l'aide des coefficients qui permettent son calcul. Ces deux représentations sont équivalentes d'un point de vue du rendu visuel, mais l'une d'elles est infiniment plus concise que l'autre. Défini dans [19] comme une image pouvant être complètement déterminée par un algorithme mathématique dans sa texture et ses détails les plus fins, on peut alors en déduire que la compression fractale consiste à obtenir une approximation d'une image réelle au moyen d'un ensemble de valeurs mathématiques.

La notion d'IFS :

La notion d'I.F.S. "Iterated Function Systems" a été inventée par le mathématicien J. Hutchinson au début des années 80. Elle désigne l'ensemble des processus itératifs qui

possèdent la propriété remarquable de converger vers un élément fixe indépendamment de leur initialisation. Le point fixe est appelé "l'attracteur" de l'I.F.S. Cette notion s'inscrit dans le cadre d'une théorie plus générale due au mathématicien d'I.B.M. Mandelbrot qui porte le nom de géométrie fractale. Cette nouvelle géométrie que l'on nomme parfois géométrie de la nature constitue une alternative au monde Euclidien pour la description de la complexité des objets naturels. Hutchinson a démontré que l'attracteur d'un I.F.S. est de type fractal, et a ainsi proposé un algorithme générateur d'images fractales.

Les Wi peuvent être décrites comme une combinaison de :

- ✓ Rotation ;
- ✓ De réduction d'échelle ;
- ✓ De translation de cordonnées des axes dans un espace à n dimension(s).

$$\omega_i = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \qquad \textbf{(1.4)}$$

Où a, b, c, d, e, et f représentent des paramètres de transformation :

$a = r\cos\theta$, $b = -s\sin\phi$, $c = r\sin\theta$, $d = s\cos\phi$

- ✓ r : facteur de réduction sur X
- ✓ S : facteur de réduction sur Y
- ✓ θ : est l'angle de rotation sur X
- ✓ ϕ : est l'angle de rotation sur Y
- ✓ e : est la translation en X
- ✓ f : est la translation en Y

Figure 1. 16 Illustre le point fixe d'une transformation affine contractive par la répétition de l'application W[12]

Par un processus itératif, utilisant des transformations géométriques affines contractives, les I.F.S. permettent de générer des images complexes. Toute l'information

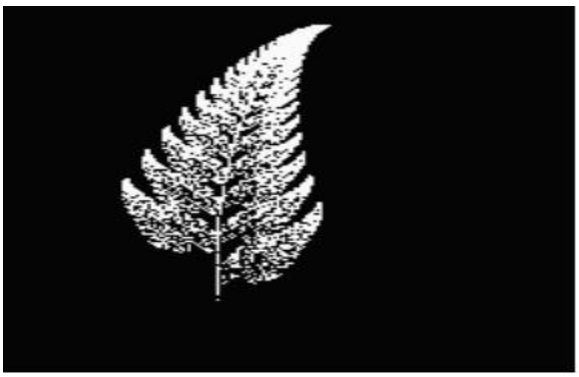
nécessaire à la formation de l'image se trouve encodée dans quelques règles simples. A titre d'exemple, l'image intitulée "la fougère de Barnsley" a été construite à partir des quatre transformations affines indiquées ci-dessous Figure 1.17.

$$\omega_1 = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.02 \\ 0.08 \end{bmatrix}$$

$$\omega_2 = \begin{bmatrix} -0.13 & 0.24 \\ 0.22 & 0.20 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.12 \\ -0.27 \end{bmatrix}$$

$$\omega_3 = \begin{bmatrix} 0.18 & -0.24 \\ 0.21 & 0.20 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -0.12 \\ -0.30 \end{bmatrix}$$

$$\omega_4 = \begin{bmatrix} 0 & 0 \\ 0 & 0.16 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ -0.42 \end{bmatrix}$$



Code IFS (1.5) **Figure 1. 17 La fougère de Barnsley**

Schéma général d'un codeur - décodeur fractal :

On peut à présent donner le schéma général (figure 1.18) [12]et l'algorithme (pseudo-code) d'un codeur et décodeur fractal générique utilisant [im98] :

• Une partition R où les blocs destinations rn = { r_i ,i= 1, n}

• Un dictionnaire où les blocs sources dα (n)

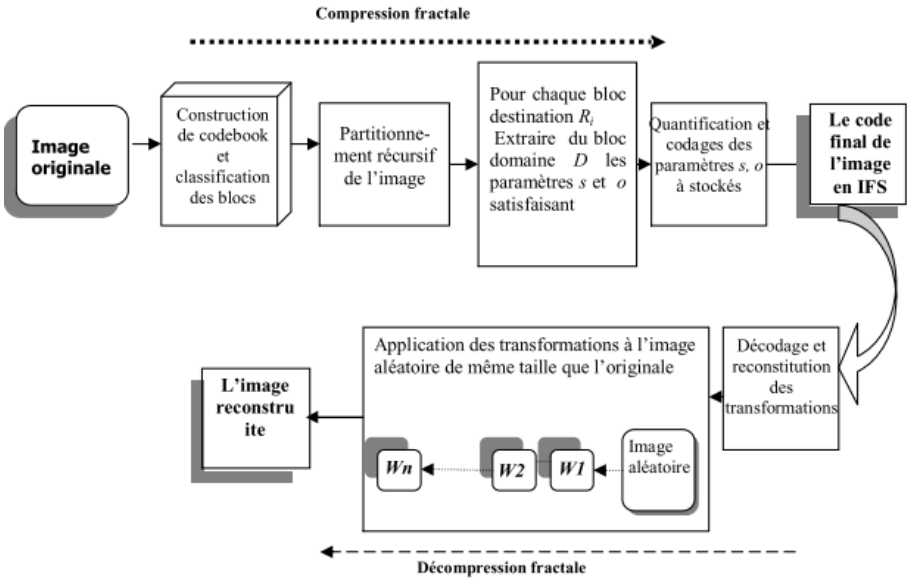


Figure 1. 18 Schéma Général d'un Codeur et Décodeur Fractale.

L'algorithme de Compression voire chapitre 02.

L'algorithme de décompression voire chapitre 02.

1.4.2.9.1 Les avantages et les inconvénients de compression fractale :

Les avantages :

- Les taux de compression élevés.
- Une représentation indépendante de l'échelle.
- Une procédure de décompression particulièrement simple, qui consiste simplement à interpréter la représentation à une échelle donnée.

Les inconvénients :

- Ce type de compression n'a pas encore fait l'objet de l'édition d'une norme, son utilisation n'est donc pas encore totalement démocratisée.
- La compression reste lente malgré toutes les améliorations (couteuse en temps) elle peut nécessite un hardware spécialise.
- Le temps très élevé pour déterminer les codes IFS de l'image compressée.
- La compression reste lente malgré toutes les améliorations (coûteuse en temps).

1.4.2.9.2 Etude comparative des différentes méthodes de compression principale :

La méthode / Propriétés	JPEG	Ondelette	Compression Fractale
Décomposition	Fréquentielle	Fréquentielle et spatial multi résolution	Limite itère d'une application contractante
Temp de compression	1s	5s	120s
Temp de décompression	1s	1s	1s
Ration pour un image de bon qualité	8	12	9
Ration pour un image de mauvais qualité	30	60	40

Tableau 1. 1 Synthèse des différents algorithmes

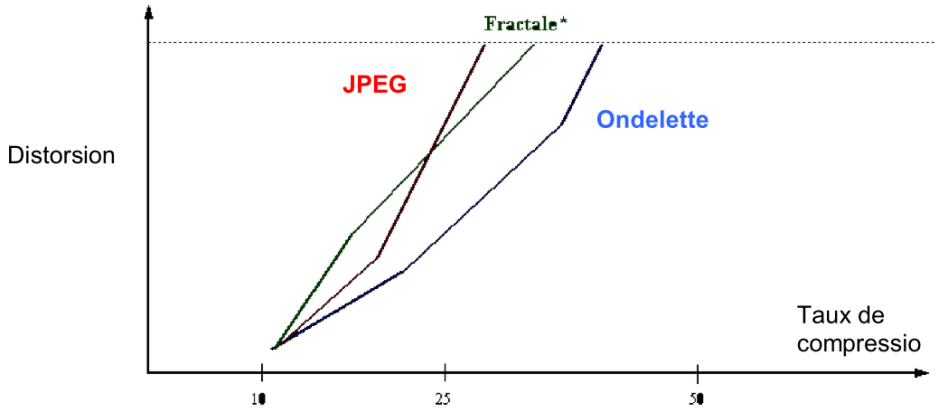


Figure 1. 19 Comparaison des méthodes en fonction (d'erreur RMSE, et taux de compression)

1.5 Métaheuristique:

Les nouveaux paradigmes ont été appelés métaheuristiques et ont été introduit pour la première fois au milieu des années 80 en tant que famille de recherche algorithmes capables d'approcher et de résoudre des problèmes complexes problèmes d'optimisation, en utilisant un

ensemble de plusieurs généraux heuristiques. Le terme métaheuristique a été proposé dans [20] définir une heuristique de haut niveau utilisée pour guider les autres heuristiques pour une meilleure évolution de l'espace de recherche. Bien que les méthodes de recherche stochastiques traditionnelles soient principalement guidées par le hasard (les solutions changent au hasard d'une étape à l'autre), ils peuvent être utilisés dans combinaison avec des algorithmes métaheuristiques pour guider le processus de recherche et d'accélérer la convergence. La plupart des algorithmes de métaheuristiques ne sont que algorithmes d'approximation, car ils ne peuvent pas toujours trouver la solution optimale globale.

Mais le plus attrayant métaheuristique est que son application nécessite aucune connaissance particulière sur le problème d'optimisation soit résolu, il peut donc être utilisé pour définir le concept de modèle général de résolution de problèmes pour l'optimisation problèmes ou autres problèmes connexes [21], [22].

Depuis leur introduction dans le milieu des années 80 jusqu'à maintenant, méthodes métaheuristiques pour résoudre l'optimisation problèmes ont été développés en permanence, permettant aborder et résoudre un nombre croissant de telles problèmes, auparavant considérés comme difficiles, voire impossible à résoudre. Ces méthodes incluent la simulation recuit, recherche tabou, calcul évolutif techniques, systèmes immunitaires artificiels, mémétique algorithmes, optimisation des essaims de particules, colonie de fourmis algorithme, évolution différentielle, recherche d'harmonie, l'optimisation des colonies d'abeilles, etc.

Les métaheuristiques (M) sont souvent des algorithmes utilisant un échantillonnage probabiliste. Elles tentent de trouver l'optimum global (G) d'un problème d'optimisation difficile (avec des discontinuités — D —, par exemple), sans être piégé par les optima locaux (L). Le schéma suivant indique Principe de métaheuristique figure 2.1.

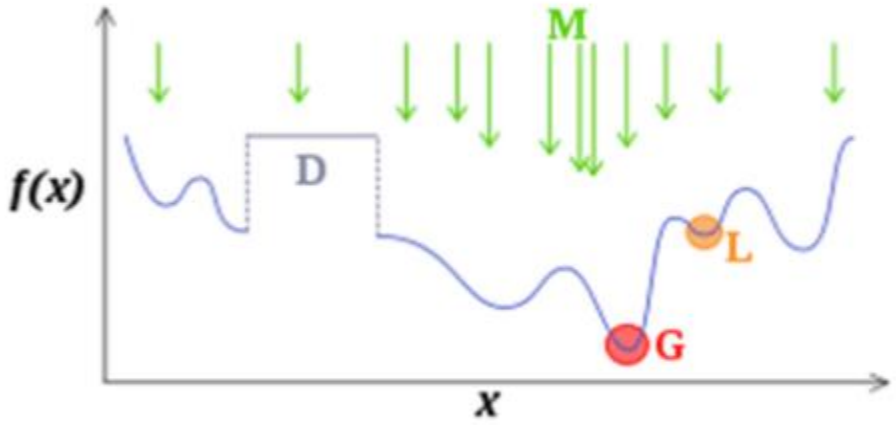


Figure 1.20 Principe de métaheuristique

1.6 Méthodes métaheuristiques :

1.6.1 Recuit simulé :

Des études sur le recuit simulé (SA) ont été développées dans les années 1980 sur la base de l'algorithme Metropolis [23], qui s'inspire de la thermodynamique statistique, où la relation entre les probabilités de deux états A et B, avec les énergies E_A et E_B, à une température commune ture T, suggère que les états aux énergies plus élevées sont moins probables dans les systèmes thermodynamiques. Ainsi, si le système est à l'état A, avec l'énergie E_A, un autre état B de niveau inférieur l'énergie $(E_B < E_A)$ est toujours possible. Inversement, un état B d'énergie supérieure ($E_B > E_A$) ne sera pas exclu, mais elle sera considérée avec la probabilité

$\exp(-(E_B - E_A) / T)$.

Application de l'algorithme Metropolis à la recherche problèmes est connu sous le nom de SA et est basé sur la possibilité de se déplacer dans l'espace de recherche vers les États les plus pauvres valeurs de la fonction fitness. À partir d'un température T et une première approximation X i, avec un fonction Fitness (X i), une perturbation est appliquée à X i générer une nouvelle approximation X i + 1, avec Fitness (X i + 1). Si X i + 1 est une meilleure solution, alors X i, c'est-à-dire Fitness (X i + 1)> Fitness (X i), la nouvelle approximation sera remplacée l'ancien. Sinon, lorsque Fitness (X i + 1) < Fitness (X i), la nouvelle approximation sera considérée avec probabilité p i = exp (- [Fitness (X i) Fitness (X i + 1)] / T).

Les étapes ci-dessus sont répétées pour un nombre donné de fois pour une valeur constante de la température, puis la température est mise à jour en diminuant sa valeur, et le processus itératif continue jusqu'à ce qu'un critère d'arrêt soit rencontré.

1.6.2 Recherche Tabu :

Tabu Search (TS) a été introduit par [20], en tant que recherche stratégie qui évite de revenir aux solutions déjà visitées en maintenant une liste de tabous, qui stocke successivement approximations.

Comme la liste de Tabu est de longueur finie, à un certain point, après plusieurs étapes, certaines solutions peuvent être revisité. Ajouter une nouvelle solution à un tabou complet la liste est faite en enlevant le plus ancien de la liste, Principe FIFO (premier entré, premier sorti).

De nouvelles approximations peuvent être générées dans différentes façons.

Parmi les nouvelles approximations le meilleur est choisi pour remplacer la solution actuelle, étant également introduite dans la liste Tabu .

1.6.3 Stratégie d'évolution :

La stratégie d'évolution (ES) a été proposée pour la première fois dans [24] en tant Que branche du calcul évolutif. C'était plus loin développé après les années 1970.

Un ES peut-être décrit par deux paramètres principaux : nombre de parents dans une génération **μ**, et nombre de descendants créés dans une génération **λ**.

La notation commune est ES (μ, **λ**). Le principal opérateur génétique qui contrôle l'évolution d'une génération à un autre est la mutation.

Dans un modèle général ES (μ**, λ**) (où **λ = k · μ**), chaque la génération commence avec une population de **λ**.

Le la condition physique de chaque individu est calculée pour les classer ordre décroissant de leur forme physique. Parmi le courant population, seuls les premiers sont plus aptes **u** individus sélectionné pour créer la population parente (cette sélection phase est parfois appelée *troncature*). Ensuite, chacun des **μ** parents vont créer par mutation répétée **k = λ / μ**

les progénitures. Finalement, la nouvelle population mutée sera remplacée l'ancien et l'algorithme réitère.

1.6.4 Algorithmes génétiques :

L'algorithme génétique (GA) est une avancée de l'ES en le cadre général du calcul évolutif.

Les GA ont été conçus et développés par Holland [25] et plus tard par Goldberg [26] et De Jong [27].

Les AG sont des stratégies de recherche basées sur des mécanismes de la génétique et de la sélection naturelle, en utilisant trois opérateurs de base : sélection, croisement et mutation.

Pour chaque génération, la sélection est utilisée pour choisir le parent individus, en fonction de leur fonction de fitness.

Après sélectionnant une paire de chromosomes parents, ils entrent dans l'étape de croisement pour générer deux progénitures. Crossover est utile pour créer de nouvelles personnes ou des solutions qui héritent bonnes caractéristiques des deux parents. Nouvellement créé les individus seront modifiés par des changements à petite échelle dans le gène, application d'un opérateur de mutation. Des mutations assurent l'introduction de "nouveauté" dans le matériel génétique.

Après compléter la population de progéniture, cela remplacera les parents de la génération précédente et le processus de sélection-croisement-mutation sera repris pour une prochaine génération. Pour ne pas perdre la meilleure solution en raison du caractère stochastique de la procédure de recherche décrite ci-dessus [27] a proposé d'appliquer un procédure de remplacement appelée «élitisme» qui fait copie de la meilleure personne de la population actuelle et le transférer inchangé dans la prochaine génération.

Le pseudocode pour l'AG est présenté dans le tableau 1.2.

Algorithmes génétiques

- Données : taille de la population **N**, taux de croisement η_c et mutation taux η_m
- Initialisation : créer la population initiale **P** = {P_i}, **i** = 1… **N**, et initialiser la meilleure solution **Best** ← **void**
- Tant que {critère d'arrêt non rempli}
  - Évalue **P** et met à jour la meilleure solution **Best**.
  - Initialiser la population de progéniture : **R** ← **vide**.
  - Créer des progénitures :

    POUR **k** = 1 À **N** / 2 faire
    - Étape de sélection : sélectionnez les parents $\mathbf{Q}_1$ et $\mathbf{Q}_2$ parmi **P**, basé sur les valeurs de fitness.
    - Étape de croisement : utiliser le taux de croisement η_c et parents (Q_1 ; $\mathbf{Q}_2$) pour créer une progéniture (S_1 **;** $\mathbf{S}_2$).
    - Stade de mutation : utiliser le taux de mutation η_m à appliquer changements stochastiques à $\mathbf{S}_1$ et $\mathbf{S}_2$ et créer des progénitures mutées $\mathbf{T}_1$ et $\mathbf{T}_2$.
    - Ajouter $\mathbf{T}_1$ et $\mathbf{T}_2$ à la population de progéniture : **R** ← **R** {T_1 et $\mathbf{T}_2$}.
  - Remplacer la population actuelle P par la progéniture population **R** : **P** ← **R.**
  - Élitisme : remplacer la solution la plus pauvre en **P** par la meilleure solution stockée dans **Best**.

Tableau 1. 2-Pseudocode pour l'algorithme génétique

1.6.5 Evolution différentielle :

Différentiel Evolution (DE) a été développé principalement par [28] comme nouvel algorithme évolutif. Contrairement à d'autres algorithmes évolutifs, DE changement successif approximations de solutions ou d'individus en fonction de la différences possibles choisies au hasard solutions. Cette approche utilise indirectement des informations à propos de la topographie de l'espace de recherche dans le quartier de la solution actuelle. Quand les solutions candidates sont choisie dans un vaste domaine, les mutations auront de grandes amplitudes.

Inversement, si les solutions candidates sont choisies dans une zone étroite, les mutations seront de peu d'importance.

Pour chaque génération, tous les individus actuels qui écrire les solutions possibles sont considérées comme référence solutions auxquelles les mécanismes de DE sont appliqués.

Ainsi, pour un individu de référence $\boldsymbol{X}_i$, deux les individus $\boldsymbol{X}_{r1}$ et $\boldsymbol{X}_{r2}$, autres que $\boldsymbol{X}_i$, sont choisis au hasard sélectionné et une mutation arithmétique est appliquée à $\boldsymbol{X}_i$ sur la base de la différence entre $\boldsymbol{X}_{r1}$ et $\boldsymbol{X}_{r2}$, pour produire un mutant $\boldsymbol{X}_i$ '. Puis un croisement arithmétique, basé sur la différence entre les solutions actuelles et mutées, est appliqué pour générer la nouvelle estimation $\boldsymbol{X}_i$ ". $\boldsymbol{X}_{je}$ " vais remplacer la solution de référence et la meilleure solution à tout moment quand sa fonction de fitness est meilleure.

1.6.6 Algorithmes immunitaires :

L'algorithme immunitaire (IA) a été proposé pour la première fois [29], simuler les capacités d'apprentissage et de mémoire de systèmes immunitaires. L'IA est une stratégie de recherche basée sur principes de l'algorithme génétique et inspirés par la protection mécanismes des organismes vivants contre les bactéries et les virus. Le problème de codage est similaire pour GA et pour IA, sauf que les chromosomes dans GA sont appelés anticorps dans l'analyse d'impact et formulation du problème, c'est-à-dire objectif ou les fonctions de fitness sont codées comme antigènes dans IA.

La différence fondamentale entre AG et IA réside dans le Procédure de sélection. Au lieu de fonctions de fitness, IA calcule les affinités entre anticorps et / ou entre anticorps et antigènes. Sur la base des affinités entre anticorps et antigènes une sélection et reproduction pool (le pool de prolifération), est créé à l'aide d'anticorps avec les plus grandes affinités. Le pool de prolifération est créé par sélection clonale : les premiers **M** anticorps, avec la plus haute affinité relative aux antigènes, sont clonées (c'est-à-dire copiées inchangé) dans le pool de prolifération. Utiliser une mutation taux inversement proportionnel à l'affinité de chaque anticorps anti-antigènes, les mutations sont appliquées à la des clones. Puis des affinités pour de nouveaux clones mutés et les affinités entre tous les clones sont calculées et un nombre limité de nombre de clones $\mathbf{N}_{rep}$ (avec les affinités les plus faibles) sont remplacés par des

anticorps générés aléatoirement, pour introduire la diversité. L'élitisme peut être appliqué pour ne pas perdre mieux solutions.

1.6.7 Optimisation des essaims de particules :

L'optimisation des essaims de particules (PSO) a été développée par Kennedy et Eberhart au milieu des années 1990 [30]. PSO est une technique d'optimisation stochastique qui émule la "Essaimage » des animaux tels que les oiseaux ou insectes. Fondamentalement, PSO développe une population des particules qui se déplacent dans l'espace de recherche à travers coopération ou interaction de particules individuelles. PSO est fondamentalement une forme de mutation dirigée.

Toute particule $\boldsymbol{i}$ est considérée en deux parties : la particule emplacement $\boldsymbol{X}_i$ et sa vitesse $\boldsymbol{V}_i$. À tout moment, le la position d'une particule $\boldsymbol{i}$ est calculée sur sa base la position $\boldsymbol{X}_i$ et un terme de correction proportionnel à sa vitesse $\boldsymbol{\varepsilon} \cdot \boldsymbol{V}_i$. A son tour, la vitesse assignée à chaque La particule est calculée en utilisant quatre composantes : (i) l'influence de la valeur précédente de la vitesse $\boldsymbol{V}_i$; (ii) l'influence de la meilleure solution personnelle pour la particule $\boldsymbol{i}$, $\boldsymbol{X}_i$ $\boldsymbol{P}$; (iii) l'influence de la meilleure solution locale jusqu'à présent pour informateurs de la particule $\boldsymbol{i}$, $\boldsymbol{X}_i$ $\boldsymbol{L}$ et (iv) l'influence de la meilleure solution globale à ce jour pour l'ensemble de l'essaim, $\boldsymbol{X}_G$. Ces composants sont pris en compte en utilisant trois facteurs de pondération, désignés par $\boldsymbol{a}$, $\boldsymbol{b}$ et $\boldsymbol{c}$. le

Le pseudocode de PSO est présenté dans le tableau 1.3.

Optimisation des essaims de particules

- Données : taille de la population **N**, poids optimal personnel **α**, population locale meilleur poids **β**, global-meilleur poids **γ**, facteur de correction **ε**.
- Initialisation : créer la population initiale **P.**
- Tant que {critère d'arrêt non rempli}
  - Sélectionnez la meilleure solution parmi les **P** actuels : **Meilleur**.
  - Choisissez la meilleure solution globale pour toutes les particules : $\mathbf{B}_{G}$.
  - Appliquer l'essaimage :

    POUR **i** = 1 À **N** faire

- Choisir la meilleure solution personnelle pour la particule $\mathbf{X}_i$: $\mathbf{B}_{iP}$.
- Choisir la meilleure solution locale pour la particule $\mathbf{X}_i$: $\mathbf{B}_{iL}$.
- Calculer la vitesse pour la particule $\mathbf{X}_i$:

  POUR **j** = 1 TO **DIM** faire

  - Générer des coefficients de correction : **a** = **α** ·rand () ; **b** = **β** · rand () ; **c** = **γ** · rand ().
  - Mise à jour de la vitesse de la particule $\mathbf{X}_i$ le long de dimension **j** :

    $\mathbf{V}_{ij} = \mathbf{V}_{ij} + \mathbf{a} \cdot (\mathbf{B}_{ij}\mathbf{P} - \mathbf{x}_{ij}) + \mathbf{b} \cdot (\mathbf{B}_{ij}\mathbf{L} - \mathbf{x}_{ij}) + \mathbf{c} \cdot (\mathbf{B}_{j}\mathbf{G} - \mathbf{x}_{ij})$

- Mise à jour de la position de la particule $\mathbf{X}_i$: $\mathbf{X}_i = \mathbf{X}_i + \boldsymbol{\varepsilon} \cdot \mathbf{V}_i$

Tableau 1.3 -Pseudocode pour l'optimisation de l'essaim de particules

1.6.8 Optimisation de la colonie de fourmis :

L'algorithme classique d'optimisation des colonies de fourmis (ACO), proposé par Marco Dorigo en 1992 [31], s'inspire du comportement naturel des fourmis, qui sont capables de trouver leur chemin en utilisant des pistes de phéromones. Voyage de fourmis entre points fixes *A* et *B* sur l'itinéraire le plus court, laissant derrière eux une traînée de phéromone deux qui marque chemins choisis. Après que la première fourmi ait atteint le point *B,* revient dans A en suivant sa propre piste de phéromone, et doubler la densité de la couche de phéromone. En outre les fourmis préféreront probablement choisir un chemin avec densité de phéromone plus élevée, et progressivement une augmentation nombre de fourmis suivront le même chemin. Pour chaque fourmi, Une liste de tabous peut être définie pour mémoriser son chemin.

Les fourmis se déplacent entre les composants C_i et C_j avec probabilité P_{ij}, définie en fonction de la densité de phéromone entre les composants, la visibilité entre les deux composantes et deux facteurs de pondération. Après chaque fourmi achève son chemin, la densité de phéromone pour chaque Le composant est mise à jour en fonction des fonctions de mise en forme.

1.6.9 Optimisation des colonies d'abeilles à miel :

Optimisation des colonies d'abeilles mellifères (HBCO) L'algorithme a été proposé pour la première fois dans [32]; c'est une recherche procédure qui imite le processus d'accouplement chez les abeilles colonies, en utilisant la sélection, croisement et mutation.

Une colonie d'abeilles à miel abrite une reine des abeilles, des drones et travailleurs. La reine des abeilles est spécialisée dans la ponte ; les drones sont les pères de la colonie et se marient avec la reine des abeilles. Pendant le vol nuptial les compagnons de reine avec des drones pour former un pool génétique. Après la génétique le bassin était rempli de chromosomes, les opérateurs génétiques sont appliqués.

Pendant la phase de croisement, les drones sont choisis au hasard dans la population et le partenaire avec la reine en utilisant un type de recuit simulé règle d'acceptation basée sur la différence entre la fonction de remise en forme du drone sélectionné et de la reine.

La dernière étape du processus évolutif consiste à élevage des couvées générées lors de la deuxième étape, et créer une nouvelle génération de couvées de $\mathbf{N}_{\mathbf{B}}$, basée sur opérateurs de mutation. Une nouvelle génération de drones $\mathbf{N}_{\mathbf{D}}$ est créé en fonction d'un critère de sélection

1.7 Hybridation de métaheuristiques :

L'hybridation consiste à combiner les caractéristiques de deux méthodes différentes pour tirer les avantages des deux méthodes [33].

Les origines des algorithmes hybrides des métaheuristiques reviennent aux travaux de Glover[34], J. J. Grefenstette [35]et Mühlenbein et al. [36].

Chacun d'eux a introduit une méthode de descente simple pour améliorer une recherche évolutive. Mais à cette période, la plupart des chercheurs n'y accordait que peu d'intérêts. Actuellement, les métaheuristiques hybrides sont devenues plus populaires car les meilleurs

résultats trouvés pour plusieurs problèmes d'optimisation combinatoires ont été obtenus avec des algorithmes hybrides. L'hybridation des métaheuristiques peut être divisée en deux grandes parties : hybridation des métaheuristiques avec des métaheuristiques et hybridation des métaheuristiques avec des méthodes exactes.

Métaheuristiques peuvent être réparties en deux catégories à savoir, les méthodes à base de voisinage correspondant à la recherche locale et les méthodes à base de population correspondant à la recherche globale comme le montre la figure 2.2.

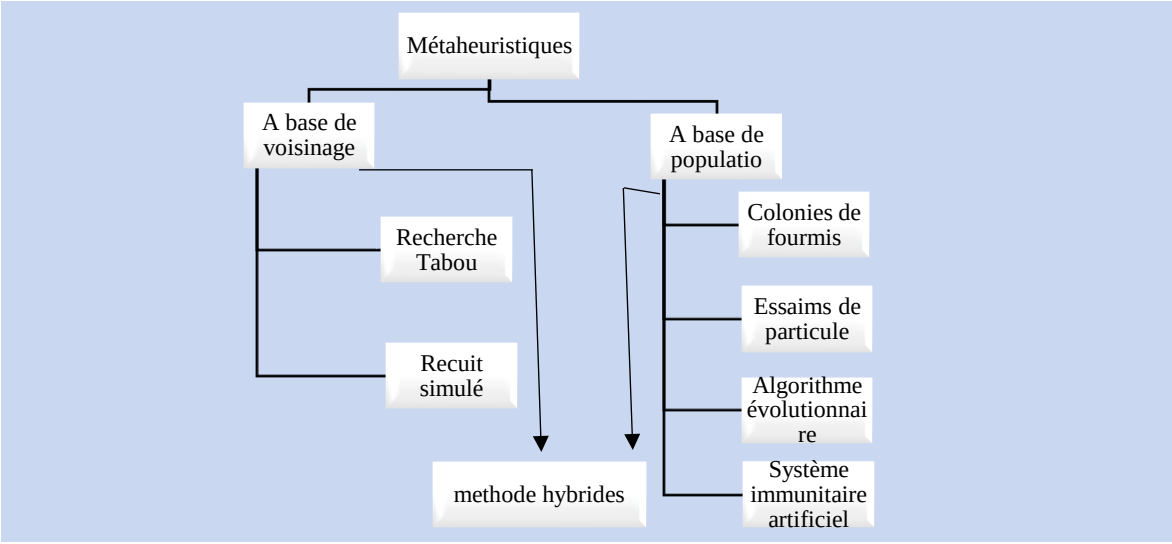


Figure 1.21 Principe générale des méthodes hybrides

1.7.1 Les algorithmes Mémétiques :

- Les algorithmes mémétiques (MA) sont une hybridation entre les algorithmes de recherche locale et les algorithmes génétiques. Les principes de ce type d'algorithme ont été introduits par Dawkins et formalisés par Moscato (Dawkins 89, Moscato 89, Moscato 99)[37]. Ils sont appelés aussi algorithmes génétiques hybrides, et Recherche locale Hybrides.

1.8 Conclusion :

La compression des données est appelée à prendre un rôle encore plus important en raison du développement des réseaux et du multimédia.

Le but de ce chapitre était d'introduire les différentes méthodes de codage entropique et de compression réversible ou irréversible des images (Huffman, LZW, …, JPEG).

La compression d'images utilisant les fractales est l'une des applications les plus récentes de la géométrie fractale.

Le chapitre suivant va introduire la différente technique, et méthode utiliser dans la compression fractale.

Chapitre 02 : Approche proposée

2.1 Introduction :

Le codage d'image fractale (FIC) basé sur le problème inverse d'un système de fonction itéré joue un rôle essentiel dans plusieurs domaines de l'infographie et dans de nombreuses autres applications intéressantes. FIC a fait l'objet d'une attention considérable en raison de sa haute résolution, de son décodage rapide et de nombreux autres avantages. Cependant, la méthode n'a pas été largement utilisée car elle nécessitait un temps de calcul élevé dans le processus de codage, ce qui est l'un de ses inconvénients. De nombreuses méthodes d'optimisation sont introduites pour résoudre cet inconvénient et réduire le temps de recherche d'une solution optimale. L'approche basée sur les méthodes métaheuristiques est prometteuse, qui emploie un certain degré de hasard pour rechercher une solution optimale.

Dans ce chapitre nous allons présenter plusieurs techniques utiliser dans la compression fractale image, De nombreux algorithmes ont été proposés pour améliorer la qualité d'image et le temps de compression.

2.2 Travaux connexes :

Dans cette section nous présentons les méthodes les plus connues dans le domaine de l'optimisation de la compression fractale d'image.

2.2.1 Hybrid Genetic-Simulated Annealing Approach for Fractal Image Compression:

Dans ce travail les auteurs[38], proposent une technique hybride d'algorithme génétique et de recuit simulé (HGASA) est appliquée à la compression d'images fractales (FIC). À l'aide de cet algorithme évolutif hybride, des efforts sont déployés pour réduire la complexité de la recherche liée à la correspondance entre bloc de plage et bloc de domaine. Le concept de recuit simulé (SA) est intégré à l'algorithme génétique (GA) afin d'éviter la convergence prématurée des chaînes. L'une des techniques de compression d'images utilisée dans le domaine spatial est la compression d'images fractales, mais le principal inconvénient de FIC est qu'elle implique plus de temps de calcul en raison de la recherche globale. Afin d'améliorer le temps de calcul ainsi que la qualité acceptable de l'image décodée, la technique HGASA a été proposée. Les résultats expérimentaux montrent que

la méthode HGASA proposée est une meilleure méthode que l'AG en termes de PSNR pour la compression d'images fractales.

2.2.2 Optimization of fractal image compression based on genetic algorithms:

Dans ce travail les auteurs [39], un « harmony search algorithm (HSA) » est appliqué à la compression fractale d'images (FIC). Cette étude introduit l'algorithme de recherche d'harmonie pour améliorer la technique FIC. L'algorithme recherche une solution optimale lors de la lecture de musique. Il a été prouvé que la recherche d'harmonie musicale résolvait les problèmes d'optimisation en recherchant une solution optimale. Cet algorithme est naturellement inspiré et est actuellement en demande. Les expériences montrent que, par rapport à d'autres techniques, la méthode proposée offre d'excellentes performances en termes de qualité d'image et réduit le temps de calcul et l'espace de stockage.

2.2.3 Fractal image compression based on spatial correlation and hybrid genetic algorithm:

Dans ce travail les auteurs[40], proposent un algorithme génétique hybride à corrélation spatiale basé sur les caractéristiques du système de fonction fractionnée et itérée (PIFS). L'algorithme comporte deux étapes : (1) Utiliser la corrélation spatiale dans les images pour le pool de domaines et le pool de domaines afin d'exploiter les optima locaux. (2) Adopter l'algorithme génétique de recuit simulé (SAGA) pour explorer les optima globaux si les optima locaux ne sont pas satisfaits. Afin d'éviter une convergence prématurée, l'algorithme adopte l'opérateur de mutation dyadique pour remplacer celui de traditionnel. Les résultats de l'expérience montrent que l'algorithme converge rapidement. Sur la base d'une bonne qualité de l'image reconstruite, l'algorithme a permis de gagner du temps de codage et d'obtenir un taux de compression élevé.

2.2.4 Fractal image compression using upper bound on scaling parameter:

Dans ce travail [41], Une approximation simple mais efficace du paramètre d'échelle est dérivée qui satisfait toutes les propriétés nécessaires pour réaliser la convergence. Cela nous permet de remplacer le processus coûteux de la multiplication matricielle par une simple division de deux nombres. Nous avons également proposé un

schéma de partitionnement de blocs horizontal-vertical (HV) modifié, ainsi que de nouveaux moyens d'améliorer le temps de codage et la qualité décodée par rapport à leurs équivalents classiques. Des expériences sur des images standard montrent que notre approche produit des performances similaires aux méthodes de compression d'images fractales les plus récentes, avec beaucoup moins de temps.

2.2.5 Crowding Optimization Method to Improve Fractal Image Compressions Based Iterated Function Systems:

Dans ce travail [42], un algorithme génétique amélioré, est utilisée pour optimiser l'espace de recherche dans l'image cible par une bonne approximation de l'optimum global en une seule analyse. Les résultats expérimentaux de la méthode proposée montrent une bonne efficacité en diminuant le temps de codage tout en conservant une image de haute qualité par rapport à la méthode classique de compression d'images fractales.

2.2.6 Hybrid Fractal Image Compression Based on Graph Theory and Equilateral Triangle Segmentation:

Dans cet article [43], les auteurs ont proposé un codage fractal basé sur la segmentation triangulaire équilatérale et ont appliqué la segmentation d'images basée sur un graphe à la compression fractale d'image, séparant l'image initiale en plusieurs zones logiques, puis codant chaque zone avec le procédé de compression d'images fractales. Les résultats expérimentaux montrent que, par rapport aux approches classiques, l'approche proposée peut accélérer la vitesse de codage fractal sur le principe de garantir la qualité des images décodées. Le schéma hybride proposé intégrant des techniques de compression fractale et d'accélération a permis d'atteindre un taux de compression élevé par rapport à la compression fractale pure.

2.2.7 Cuckoo inspired fast search algorithm for fractal image encoding:

Dans cet travaille les auteurs [44], les auteurs ont proposé une technique de recherche rapide inspirée par le coucou (CIFS) pour la compression fractale d'images.

Contrairement aux nombreux modèles traditionnels, qui dépendent d'une classification en ondelettes à 3 niveaux, le CIFS proposé utilise un vecteur ordonné de blocs de distance par leur similarité et un vecteur ordonné de blocs d'intervalle par leur distance de coordonnées. L'étude expérimentale a démontré que le modèle proposé est évolutif et robuste par rapport aux modèles basés sur PSO et GA que l'on trouve dans la littérature contemporaine. La réduction significative du calcul de l'erreur quadratique moyenne est également observée, car les quatre seules transformations du groupe dihédral sont suffisantes pour permettre la comparaison des similitudes dans ce CIFS proposé.

2.2.8 A fractal image encoding method based on statistical loss used in agricultural image compression:

Dans ce travail les auteurs [45], ont utilisé d'abord l'analyse statistique du procédé de codage fractal. Ensuite ils ont créé une boîte à moustaches pour trouver la distribution de la valeur de perte. Ensuite, nous les partitionnons en plusieurs parties et les mappons au modèle donné. Après cela, nous présentons une nouvelle méthode pour économiser la perte et maintenir la qualité de la compression d'image. Enfin, les résultats expérimentaux agricoles montrent l'efficacité de la nouvelle méthode.

2.2.9 Fractal image coding algorithm using particle swarm optimization and hybrid quadtree partition scheme:

Dans ce travail les auteurs [46], proposent un nouvel algorithme de codage fractal d'image utilisant l'optimisation par essaim de particules (PSO) et le schéma de partition hybride à quatre arbres (QP) est proposé. Une méthode appelée stratégie PSO basée sur la classification par bloc de distance (PSO-RC) est présentée au lieu d'utiliser la méthode PSO dans le pool de plage entier. Cette nouvelle idée peut améliorer considérablement le taux de compression et accélérer le codeur. De plus, un schéma QP hybride PSO-RC (PSO-RCQP) est adopté afin d'améliorer davantage la qualité de l'image extraite. Premièrement, les blocs de plage sont divisés en deux catégories sur la base de la caractéristique de déviation standard. Deuxièmement, les blocs de plage sont codés en utilisant l'approche PSO ou en stockant directement les valeurs de pixel moyennes. Troisièmement, les blocs de plage avec des erreurs de correspondance importantes lors de l'utilisation du schéma

PSO utilisent le procédé QP proposé. Les résultats de la simulation montrent que l'algorithme proposé peut obtenir une bonne qualité et un taux de compression plus élevé que le raccourcissement du temps de codage.

2.2.10 Fractal image compression based on spatial correlation and hybrid particle swarm optimization with genetic algorithm:

Dans ce travail les auteurs [47], proposent un algorithme de compression d'images fractales basé sur la corrélation spatiale et l'optimisation des essaims de particules hybrides avec un algorithme génétique (SC-PSOGA) est proposé pour réduire l'espace de recherche. L'algorithme comporte deux étapes : (1) Utiliser la corrélation spatiale dans les images pour le pool de domaines et le pool de domaines afin d'exploiter les optima locaux. (2) Adopter l'optimisation des essaims de particules hybrides avec l'algorithme génétique (PSO-GA) afin d'explorer les optima globaux si les optima locaux ne sont pas satisfaits. Les résultats de l'expérience montrent que l'algorithme converge rapidement. Au principe de bonne qualité de l'image reconstruite, l'algorithme enregistre le temps d'encodage et obtenu un taux de compression élevé.

2.2.11 An edge property-based neighborhood region search strategy for fractal image compression:

Dans ce travail [48], une méthode de recherche de région de voisinage basée sur les propriétés de bord est proposée pour accélérer le codeur fractal. La méthode recherche la solution la mieux adaptée dans le domaine fréquentiel. Un système de coordonnées est construit à l'aide des deux plus bas coefficients de transformation en cosinus discret (DCT) des blocs d'image. Les blocs d'image ayant des formes d'arêtes similaires seront concentrés dans certaines régions spécifiques. Par conséquent, l'objectif d'accélération peut être atteint en limitant l'espace de recherche. De plus, en incorporant la propriété de bord du bloc dans le procédé de recherche proposé, le taux d'accélération peut être encore augmenté. Les résultats expérimentaux montrent que, dans les conditions d'un même PSNR, le temps de codage de la méthode proposée n'est que d'environ deux cinquièmes de la méthode de classification de Duh. Comparée à la méthode de Tseng, la méthode proposée est proche

ou supérieure à la performance de leur méthode. De plus, la vitesse d'encodage de la méthode proposée est environ 120 fois plus rapide que celle de la méthode de recherche complète, tandis que la pénalité liée à la qualité d'image obtenue ne décroît que de 0,9 dB.

2.2.12 Introducing BAT inspired algorithm to improve fractal image compression:

Dans ce travail [50], nous appliquons et pour la première fois, une nouvelle étude sur les performances de l'algorithme BATInspired (BIA) en tant que méthode naturelle inspirée de l'histoire de l'homme pour améliorer FIC. Les résultats montrent l'amélioration de cet algorithme sous différents aspects (temps de codage, taux de compression (CR), rapport signal / bruit de crête (PSNR) et erreur quadratique moyenne (MSE)). En outre, une comparaison avec certaines des méthodes actuelles est jugée moins satisfaisante.

2.2.13 An improved fractal image compression using wolf pack algorithm :

Dans ce travail [51], et pour la première fois, une étude plus détaillée sur l'algorithme de Wolf Pack pour la compression d'images fractales. L'ensemble de l'image est considéré comme une recherche dans l'espace où cet espace est divisé en blocs, les loups scooters explorent cet espace pour trouver un autre bloc plus petit qui présente une similitude avec ses paramètres. Les loups scooters parcouraient tout l'espace et sélectionnaient les blocs avec la meilleure condition physique. Le processus sera arrêté après un nombre fixe d'itérations ou si aucune amélioration de la solution de loup de plomb. Les résultats montrent que, comparée à la méthode de recherche exhaustive, la méthode proposée a considérablement réduit le temps de codage et obtenu un taux de compression plutôt optimal. Les expériences réalisées ont montré son efficacité dans la résolution de ce problème. De plus, une brève comparaison avec les différentes méthodes établit cet avantage**.**

2.3 Approche proposée :

2.3.1 Algorithme PSO-GA :

Chaque particule a une vitesse et une position qui sont représentées respectivement par les deux vecteurs : X et Y avec 4 dimensions. Chaque particule X est un vecteur de

4dimensions représentant une combinaison des variables : a, b, c, k. Par conséquent, le but de PSO est de trouver la meilleure combinaison des 4 paramètres, de l'équation (1),

Particule X=

a	b	c	k

Représentation d'une particule (1).

1. Générer la population initiale avec des positions aléatoires des particules.

2. Initialiser la meilleure position personnelle (pi) de chaque particule à sa position initiale.

3. Pour chaque itération :

a) Transformer l'image originale selon des valeurs de a, b, c et k données par chaque particule i en utilisant l'équation (1)

b) évaluer le fitness de chaque particule (basée sur l'image transformée correspondante) utilisant la fonction objective décrite dans l'équation

Fonction fitness=MSE (Mean Square Error).

c) trouver la meilleure position globale dans l'essaim.

d) mettre à jour la position et la vitesse de chaque particule en utilisant les équations (2.1).

$$\begin{cases} v_{ij}(t)=wv_{ij}(t-1)+\varphi_{1ij}(p_{ij}-x_{ij}(t-1))+\varphi_{2ij}(p_{gj}-x_{ij}(t-1)) \\ x_{ij}(t)=x_{ij}(t-1)+v_{ij}(t) \end{cases} \quad (2.1)$$

4. Trouver la meilleure position dans l'essaim,

5. Générer une population initiale. (Solution d'algorithme PSO).

6. Pour chaque itération

a) Appliquer l'opération de sélection (basée sur l'évaluation de la fitness),

function fitness=MSE (Mean Square Error).

b) Appliquer l'opération de croisement sur le résultat obtenir par l'opération de sélection.

c) Appliquer l'opération de mutation sur le résultat obtenir par l'opération de croisement.

d) Appliquer l'opération de sélection sur le résultat obtenir par l'opération de mutation

7. fin d'itération stop de critères.

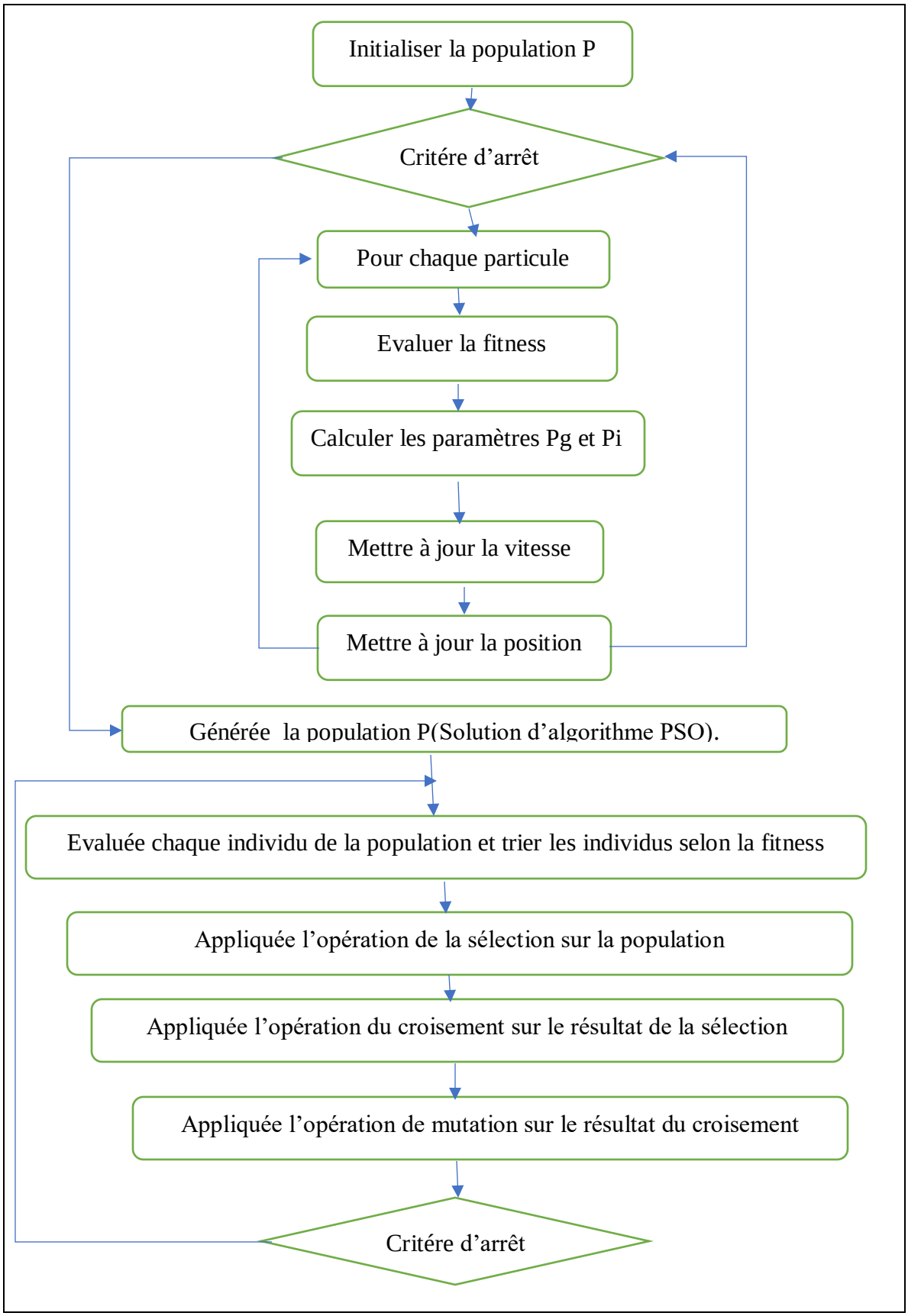

Initialiser la population P
Critére d'arrêt
Pour chaque particule
Evaluer la fitness
Calculer les paramètres Pg et Pi
Mettre à jour la vitesse
Mettre à jour la position
Générée la population P(Solution d'algorithme PSO).
Evaluée chaque individu de la population et trier les individus selon la fitness
Appliquée l'opération de la sélection sur la population
Appliquée l'opération du croisement sur le résultat de la sélection
Appliquée l'opération de mutation sur le résultat du croisement
Critére d'arrêt

Figure 2.1 Algorithme PSO-GA .

2.3.2 Algorithme Fractal :

2.3.2.1 L'algorithme de Compression :

Etape 1 : Lire l'image originale

Etape 2 : Fixer le seuil (l'erreur RMS)

Etape 3 : Donner Tmin, Tmax (taille min et Max du bloc à partitionner)

Etape4 : Construction du dictionnaire des blocs sources dα(n) (domaines) et leurs classifications.

Etape 5 : Partitionnement (quadtree) et récursif de l'image.

Etape 6 : Pour chaque bloc destination (range ri (i=1……n))

- Classifier le bloc destination ri
- Rechercher dans la classe du bloc destination le bloc source distorsion calculée < seuil fixé RMS (l'erreur quadratique) dα qui satisfait la (n)
- Si oui stocker les paramètres de la transformation Wi du bloc destination luminance : o, contraste : s, les coordonnées du bloc source, …
- Si non (aucun bloc ne satisfait la distorsion calculée < seuil RMS)
    - Si (la taille du bloc > Tmin alors) décomposer le bloc destination à nouveau
    - Si non la taille du bloc destination inférieure à la taille Tmin. En prend le bloc source (domaine) avec la plus faible erreur possible et son paramètre sera stocker (Wi du bloc destination luminance : o, contraste : s, les coordonnées du bloc source)

Etape 7 : Quantification des paramètres stockés et codage

Etape 8 : Fermer la bibliothèque domaine

Etape 9 : Terminer le processus de compression (résultat est un fichier IFS).

2.3.2.2 L'algorithme de décompression :

Etape 1 : Lire ou ouvrir le fichier IFS

Etape 2 : Décodage et reconstitution des transformations

Etape 3 : Charger une image A aléatoire de même taille que l'image originale à compresser

Etape 4 : Pour chaque transformation Wi (i=1, N)

• Appliquer la transformation Wi à l'image A

• Reconstruire l'image A

• Passer à la transformation suivante (aller à l'étape 4)

Etape 5 : Traitement et Affichage de l'image résultant

2.3.3 Méthode Proposée :

Algorithme Fractale PSO-GA :

1-Demarre l'algorithme de compression fractale d'image.

2-demarre l'algorithme PSO-GA.

3-Demarre l'algorithme de décompression fractale.

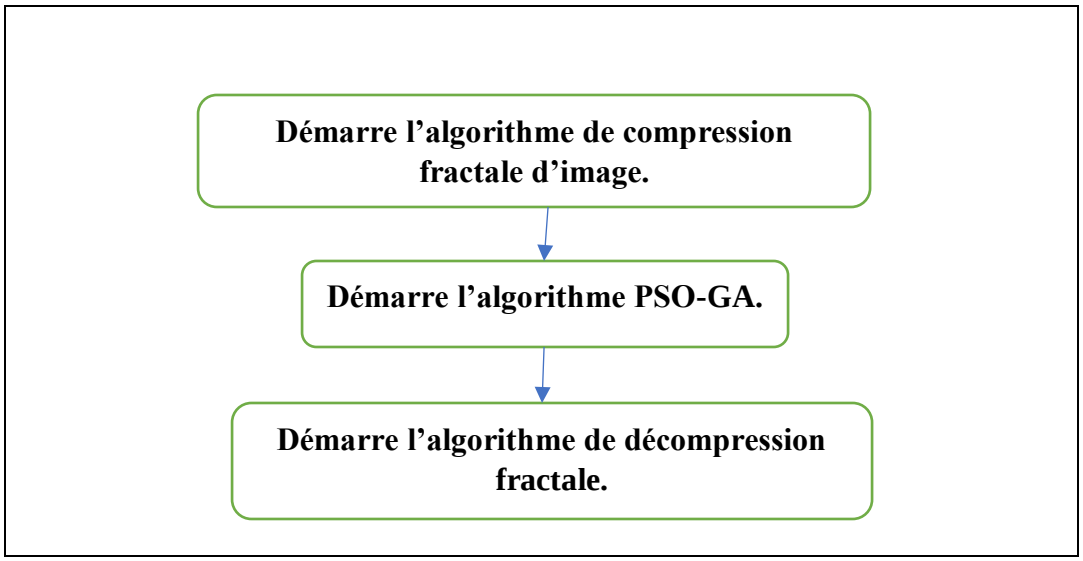


Figure 2.2 Algorithme Fractale PSO-GA

2.4 Conclusion :

Dans ce chapitre on a essayé de présenter brièvement les travaux similaires les plus récents dans l'optimisation de la compression fractale d'image. Alors on a essayé de donner plus de détails, et une vue d'ensemble, sur les méthodes d'optimisation proposées jugées récentes et importants et qui sont plus proches de notre méthode, et qui, dans l'ensemble ont pour but, l'amélioration de qualité d'image et la réduction du temps de compression.

Ce qu'on peut conclure après cette étude, est la puissance que cette famille d'algorithmes dans la résolution des problèmes d'optimisation complexes et difficiles, aussi on a remarqué une simplicité relative dans l'adaptation de ces méthodes pour ce genre de problèmes.

Chapitre 03 : Application d'une Hybridation (PSO-GA) à la compression fractale d'images

3.1. Introduction :

Pour implémenter une hybridation entre les algorithmes génétique et les algorithmes des PSO et l'appliquer à la compression fractale, on a choisi d'utiliser l'environnement de programmation Matlab parce qu'il offre un Toolbox complet de traitement d'images qui permet de faciliter à l'utilisateur de traiter ses images.

Les algorithmes GA, PSO sont utilisés comme des techniques hybride pour réduire le temps de compression d'image local ayant les objectifs suivants :

a) ne pas utiliser l'interaction avec l'utilisateur, pendant les étapes courantes de l'algorithme, donc la méthode est complètement automatique.

b) utiliser un critère objectif de fitness sans des paramètres externes additionnels, On appliquer AG car les objectifs suivants.

Elle peut parcourir rapidement un grand ensemble de solutions, La nature inductive des AG signifie qu'il ne doit connaître aucune règle du problème. Elle travaillée avec leurs propres règles internes.

3.2 Rapport et taux de compression :

Le rapport de compression est l'une des caractéristiques les plus importantes de toutes les méthodes de compression, il représente le rapport entre le nombre de bits de la forme canonique au nombre de bits après codage :

$$\text{Rapport}_c = \text{CR} = \frac{\text{nombre de bits de l image avant compression}}{\text{nombre de bits de l image apr}} = \frac{R_0}{R_c} \qquad (3.1)$$

Par conséquent le taux de compression est un pourcentage de l'espace obtenu après la compression par rapport à l'espace total requis par les données avant la compression. Il est défini par :

$$T_c = \left(1 - \frac{1}{\text{rapport de compression}}\right) * 100 \qquad \textbf{(3.2)}$$

3.3 Mesure de la qualité visuelle de l'image reconstruite

La mesure de la performance d'un codeur en termes de qualité visuelle de l'image reconstruite se fait généralement en mesurant le rapport signal a bruit ou le rapport signal a bruit de crête dont les expressions sont rappelées ci-dessous :

Le rapport signal a bruit, note SNR, entre l'image originale composée de pixels x (m n) et l'image décode Composer de pixels x' (m n) est donne par :

$$SNR = 10\log_{10}\frac{E\left[x\left(m,n\right)^2\right]}{E\left[\left(x\left(m,n\right)-\hat{x}\left(m,n\right)\right)^2\right]}$$
$$= 10\log_{10}\frac{\sum_m\sum_n\left(x\left(m,n\right)\right)^2}{\sum_m\sum_n\left(x\left(m,n\right)-\hat{x}\left(m,n\right)\right)^2} \quad \textbf{(3.3)}$$
$$= 10\log_{10}\frac{E\left[x\left(m,n\right)^2\right]}{MSE}$$

Le rapport signal a bruit de crête, note PSNR, (plus souvent utilise en compression) est donne par :

$$PSNR = 10\log_{10}\frac{255^2}{E\left[\left(x\left(m,n\right)-\hat{x}\left(m,n\right)\right)^2\right]} \quad \textbf{(3.4)}$$

Ces mesures sont très utilisées car très simples à calculer. Ce ne sont cependant pas des mesures ables de ressemblance visuelle entre deux images [49]. Un contre-exemple consiste à prendre deux images composées d'un bruit blanc. L'erreur MSE est très grande entre les deux images alors qu'elles sont très semblables visuellement.

3.4 L'environnement de travail :

3.4.1 Le système d'exploitation :

L'environnement WINDOWS 10 PRO a été choisi comme environnement de travail pour notre logiciel pour les raisons suivantes :

- Une très bonne gestion de mémoire ;
- Une architecture orientée évènement ;
- Un graphisme indépendant des périphériques ;
- La notion de ressources.

3.4.2 Ressources matérielles :

Notre travail s'est effectué sur un micro-portable ayant les caractéristiques suivantes :

- Processeur Intel (R) Core² (TIM) i3-320M CPU @ 2,50 GHZ.
- 4 GO de RAM ;
- Ecran couleur 15,6 pouces.

3.4.3 Langage de programmation :

A l'origine, la toute première version de Matlab fut mise en œuvre aux Universités du Nouveau Mexique et de Stanford, à la fin des années 70. Elle se destinait essentiellement à l'analyse numérique et à l'algèbre linéaire. Depuis, Matlab est devenu un outil que l'on peut utiliser pour mener des simulations dans de très nombreux domaines (traitement du signal, traitement d'image, automatique, statistique, aéronautique, etc.)

3.4.4 Matlab et ses bibliothèques :

Matlab est un logiciel qui dispose d'une collection de fonctions/programmes, que l'on désigne aussi par des fichiers à extension.m. Ces .m sont classés par thèmes et constituent alors des boîtes à outils (ou toolboxes en anglais). On dispose ainsi des toolboxes 'général', 'identification', 'signal', etc. Matlab ayant été créé à partir d'une bibliothèque dédiée au calcul matriciel, de nombreuses bibliothèques sont relatives à l'algèbre linéaire ('elmat', 'specmat', 'matfun').

Ces bibliothèques sont mises en place au moment de l'installation de Matlab. Il est à noter qu'elles sont payantes. Mais elles sont très utiles car elles regroupent des traitements de bases (par exemple, le calcul du déterminant d'une matrice, l'obtention des valeurs propres ou singulières et des vecteurs associés, etc.). En général, chaque programme débute par un commentaire de quelques lignes décrivant l'objectif du programme, les entrées, etc.

Comme les bibliothèques doivent être accessibles de n'importe quel point où l'on se place dans l'arborescence du disque, une variable PATH doit être configurée pour définir les chemins d'accès. L'ensemble des liens vers des bibliothèques existantes est disponible en tapant dans la fenêtre de commandes : path.

3.4.5 Description de la fenêtre de commande de Matlab :

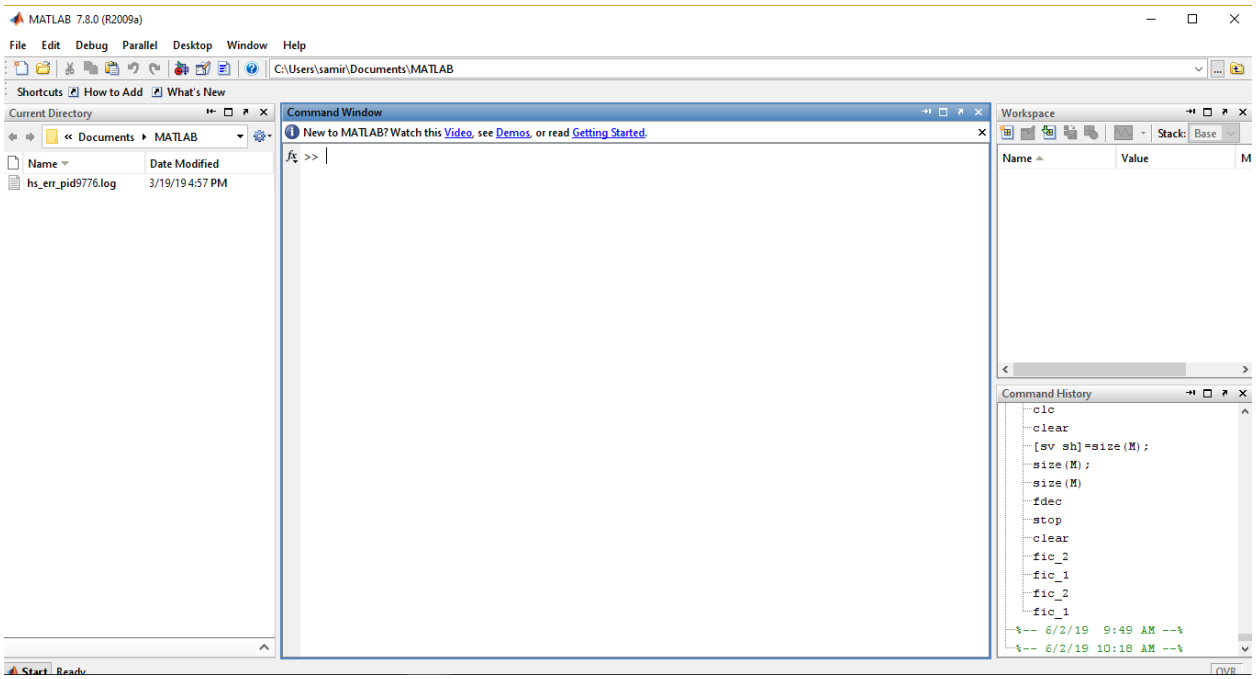


Figure 3.1 la fenêtre de commande de Matlab

Il existe quatre menus déroulants :

- **File** : le menu fichier permet notamment d'accéder à des fichiers déjà existants ou de créer de nouvelles procédures et des nouveaux programmes. Il permet aussi de quitter Matlab. Les derniers programmes ouverts sont en outre indiqués.

- **Edit** : la menue édition permet d'accéder aux commandes classiques : copier, couper, coller, sélectionner tout, etc.

- **Window** : le menu Fenêtre donne accès aux différentes fenêtres de Matlab ouvertes.

- **Help** : c'est le menu d'aide.

Le soulignement des lettres F, E, W et H des menus File, Edit, Windows, Help indique le raccourci clavier pour ouvrir le menu déroulant sans l'aide de la souris. On effectue alors la commande alt+lettre soulignée. Comme dans les logiciels Word et Excel, on peut aussi exploiter la barre d'outils. Cette dernière comprend plusieurs « boutons » qui permettent notamment de créer une nouvelle.

3.5 Formulation du problème par PSO-GA :

Le codage d'image fractale (FIC) basé sur le problème inverse d'un système de fonction itéré joue un rôle essentiel dans plusieurs domaines de l'infographie et dans de nombreuses autres applications intéressantes. FIC a fait l'objet d'une attention considérable en raison de sa haute résolution, de son décodage rapide et de nombreux autres avantages. Cependant, la méthode n'a pas été largement utilisée car elle nécessitait un temps de calcul élevé dans le processus de codage, ce qui est l'un de ses inconvénients. De nombreuses méthodes d'optimisation sont introduites pour résoudre cet inconvénient et réduire le temps de recherche d'une solution optimale. L'approche basée sur les méthodes métaheuristiques est prometteuse, qui emploie un certain degré de hasard pour rechercher une solution optimale.

Cette étude introduit l'algorithme hybride (PSO-GA) pour améliorer la technique FIC. Des algorithmes hybrides avec GA et PSO sont proposés [47]. La base derrière cela est qu'une telle approche hybride devrait avoir des avantages de PSO avec ceux de GA. Les systèmes PSO hybrides proposés trouvent une meilleure solution sans emprisonner le maximum local et obtenir un taux de convergence plus rapide. En effet, lorsque les particules de PSO stagnent, GA diversifie la position des particules même si la solution est pire. Dans PSO-GA, le mouvement des particules utilise le hasard dans sa recherche. Il s'agit donc d'une sorte d'algorithme d'optimisation stochastique capable de rechercher une zone complexe et incertaine. Cela rend PSO-GA plus flexible et robuste.. Les expériences montrent que, par rapport à d'autres techniques, la méthode proposée offre d'excellentes performances en termes de qualité d'image et réduit le temps de calcul et l'espace mémoire.

3.6.1 Application De Hybride (PSO-GA) Au compression fractale D'images :

On se rappelle des formules utilisées pour la mise à jour des valeurs des positions et celles des vitesses des particules dans l'espace de recherche dans le cas du PSO :

$$\begin{cases} v_{ij}(t) = w v_{ij}(t-1) + \varphi_{1ij}(p_{ij} - x_{ij}(t-1)) + \varphi_{2ij}(p_{gj} - x_{ij}(t-1)) \\ x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \end{cases} \quad (3.5)$$

Où :

v(t) : est la vitesse courante ;

x(t) : est la position courante ;

p(t) : est sa meilleur position précèdent ;

g(t) : est la meilleur position trouvée par ces voisin ;

$\varphi_{1ij} = c_1 * r_{1ij}$

et

$\varphi_{2ij} = c_2 * r_{2ij}$ et c_1*et* c_2 sont les constant :cognitive et social d'accélération, et r_{1i} et r_{2i} *sont* des nombre aleatoire distribués entre 0 et 1 w estla poids d'inertie.

3.6.2 Choix des paramètres :

Pour appliquer les deux modèles d'optimisation PSO et GA à la compression fractale d'image, on doit indiquer les valeurs des paramètres utilisés pour la mise à jour des positions et des vitesses des particules dans l'essaim PSO, et les nombre d'itération et nombre de population (voir le tableau 3.1.) donc, on doit indiquer les valeurs des paramètres employés par PSO (voir le tableau 3.2.) :

Tableau 3.1 Les paramètres d'hybride (PSO-GA).

Méthode		Taille de la population	Itération
PSO-GA	PSO	5	5
	GA	5	5

Tableau 3.2 Les paramètres de l'algorithme PSO.

Paramètres	Valeur
wmi n	0.4
wmi x	0.9
Constant social c1	1.3
Constant cognitive c2	1.4

3.6.2 Résultats et discussion :

Image lina.bmp (256*256)

Tableau 3.3 Résultat de lina .

		Temp de compression (s)	Temp de décompression (s)	Compression de Ration	PSNR
	Full search	1777.888219 seconds.	31.406472 seconds.	1	30.33
	Méthode proposée (PSO-GA)	110.634058 seconds	30.633901 seconds	1	25.08

Image barbara.jpg (256*256)

Tableau 3.4 Résultat de barbara.

		Temp de compression (s)	Temp de décompression (s)	Compression de Ration	PSNR
	Full search	1745.161544 seconds.	33.144218 seconds.	1	31.61
	Méthode proposée (PSO-GA)	110.748747 seconds.	31.562786 seconds	1	26.02

Image cameraman.jpg (256*256)

Tableau 3.4 Résultat de cameraman.

		Temp de compression (s)	Temp de décompression (s)	Compression de Ration	PSNR
	Full search	1805.354383 seconds.	34.685817 seconds	1	29.68
	Méthode proposée (PSO-GA)	117.852125 seconds.	30.532819 seconds.	1	24.06

Résolution d'image linna.bmp (32*32) (64*64) (128*128)

Tableau 3.5 Résultat de résolution linna.

(32*32)		Temp de compression (s)	Temp de décompression (s)	Compression de Ration	PSNR
	Full search	0.720017 seconds.	0.558404 seconds.	1	41.07
	Méthode proposée (PSO-GA)	0.132515 seconds.	0.541288 seconds.	1	35.82
(64*64)	Full search	6.973871 seconds.	2.069901 seconds.	1	38.96
	Méthode proposée (PSO-GA)	0.728786 seconds.	1.933325 seconds.	1	32.46
(128*128)	Full search	148.500767 seconds.	12.014393 seconds.	1	35.20
	Méthode proposée (PSO-GA)	10.110445 seconds.	9.739570 seconds.	1	29.83

Tableau 3.6 Comparaison la méthode proposer avec d'autre méthode :

Image	Méthode	PSNR(dB)	Temp de compression (s)	Compression de Ration
Lena128*128	PSO-GA	29.83	10.110	1
	BIA(BatInspired Algorithm)[50]	32.909	33.376	1.486
	Suman K. Mitra etal.[57]	30.22	/	1.059
	VishvasV.Kalunge etal.[53]	/	67	/
Lena256*256	PSO-GA	25.08	110.634	1
	BIA[50]	33.115	732.345	1.604
	Y.Chakrapanietal. [56]	26.22	2340	1.3
	Exhaustivesearch	32.69	8400	1.3
	DWSR[53]	25.8212	56.4247	1.56355
	PSO-RCQP[54]	27.089	6.453	1.6392
	AFIC-PSO[55]	35.03	934	/
Cameraman 256*256	PSO-GA	24.06	117.852	1
	BIA[50]	31.095	732.011	1.678
	PSO-RCQP[54]	26.686	268	1.8212
	AFIC-PSO[55]	34.23	924	/

Conclusion générale et perspective :

Dans notre travail Nous avons appliqué des méthodes d'optimisation hybride (algorithme génétique) AG et PSO (Particle Swarm Optimization), pour améliorer la qualité d'image et réduire le temps de calcul.

Un ensemble d'images en niveaux de gris a été employé pour prouver l'efficacité de notre approche proposée pour réduire d'une façon remarquable le temps de compression. On a montré dans le dernier chapitre, les résultats obtenus en appliquant l'hybridation de l'algorithme génétique et PSO pour chaque image avec les autres résultats obtenus en calculant les deux mesures PSNR et MSE.

Il était clair que les résultats obtenus par l'application de l'hybridation Génétique et PSO sont normale par rapport à ceux de la technique classique de compression d'image fractale.

Nous envisageons dans un futur travail d'autres hybridations des méthodes d'optimisation pour la compression fractale et d'autres méthodes de compression. Et pourquoi pas essayer les méthodes perturbatrices.

Référence

[1] J. Desachy, “Analyse d’images,” *notes de cours-version*, vol. 1, 2001.

[2] M. Šonka, V. Hlavác, and R. Boyle, *Image processing, analysis, and machine vision*. Thomson, 1993.

[3] “An Analysis of Evolutionary Computation used in Image Processing Techniques John P . Cartlidge BSc Artificial Intelligence and Mathematics 1999 / 2000 Summary,” 2000.

[4] N. MOUNIB, “Une Approche Co-évolutionnnaire Pour Le Réhaussment D’images.” Université de Batna 2, 2007.

[5] Y. Gagou, “Cours de traitement d’image,” *Univ. Picardie Jules Verne*, 2008.

[6] W.-Y. Wei, “An Introduction to Image Compression.” pp. 1–29, 2013.

[7] S. C. Hinds, J. L. Fisher, and D. P. D. Amato, “7525 Usa,” *Neuropsychology*, pp. 464–468, 1990.

[8] D.-H. Xu, A. S. Kurani, J. D. Furst, and D. S. Raicu, “Run-Length Encoding for Volumetric Texture,” *Heart*, vol. 27, no. 25, pp. 452–458, 2004.

[9] D. A. Huffmant, “Minimum-Redundancy Codes *,” *Proceeding of the I.R.E*, vol. 27, pp. 1098–1101, 1951.

[10] J. Ziv and A. Lempel, “A Universal Algorithm for Sequential Data Compression,” *IEEE Trans. Inf. Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[11] G. MERCIER, C. ROUX, and G. MARTINEAU, “Technologie de Multimédia,” *ENST Bretagne, dpt ITI, BP832, F-29280 Brest, Fr.*, vol. 15, 2003.

[12] Z. Djazia and M. Benmohammed, “Impl??mentation d’un environnement parall??le pour la compression d’images ?? l’aide des fractales,” *Proc. Int. Conf. Comput. Sci. its Appl.*,

2006.

[13] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, vol. 159. Springer Science & Business Media, 2012.

[14] A. K. Jain, *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[15] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 36–58, 2001.

[16] S. Renard, "Club Photoshop de Nantes Conférence du 14 octobre 1999 Résumé et compléments."

[17] C. Delgorge, "Proposition et évaluation de techniques de compression d'images ultrasonores dans le cadre d'une télé-échographie robotisée." Orléans, 2005.

[18] D. Santa Cruz, R. Grosbois, and T. Ebrahimi, "JPEG 2000: la nouvelle norme pour le codage d'images." organisés par le Service informatique central de l'EPFL, Sommaire FI3, 2001.

[19] P. A. Зубко, "Fractal image compression method," *Eastern-European J. Enterp. Technol.*, vol. 6, no. 2(72), p. 23, 2014.

[20] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986.

[21] K. Y. Lee and M. A. El-Sharkawi, *Modern heuristic optimization techniques: theory and applications to power systems*, vol. 39. John Wiley & Sons, 2008.

[22] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, 1983.

[24] I. Rechenberg, "Cybernetic solution path of an experimental problem," *R. Aircr. Establ. Libr. Transl. 1122*, 1965.

[25] J. H. Holland, “Adaptation in natural and artificial systems Ann Arbor,” *Univ. Michigan Press*, vol. 1, p. 975, 1975.

[26] D. E. Goldberg, “Genetic algorithms in search,” *Optim. Mach.*, 1989.

[27] K. A. De Jong, “Genetic algorithms: A 10 year perspective,” in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, 1985, vol. 1, no. 6, pp. 9–177.

[28] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[29] J. D. Farmer, N. H. Packard, and A. S. Perelson, “The immune system, adaptation, and machine learning,” *Phys. D Nonlinear Phenom.*, vol. 22, no. 1–3, pp. 187–204, 1986.

[30] J. Kennedy and R. Eberhart, “Particle Swarm Optimization. proceddings of IEEE International Conference on Neural Networks,(pp. 1942–1948),” *Perth, WA, Aust.*, 1995.

[31] M. Dorigo, “Optimization, learning and natural algorithms,” *PhD Thesis, Politec. di Milano*, 1992.

[32] S. Nakrani and C. Tovey, “On honey bees and dynamic server allocation in internet hosting centers,” *Adapt. Behav.*, vol. 12, no. 3–4, pp. 223–240, 2004.

[33] K. A. Publishers and G. Talbi, “A Taxonomy of Hybrid Metaheuristics,” pp. 541–542, 2002.

[34] F. Glover, “Heuristics for integer programming using surrogate constraints,” *Decis. Sci.*, vol. 8, no. 1, pp. 156–166, 1977.

[35] J. J. Grefenstette, “Incorporating problem specific knowledge in genetic algorithms,” *Genet. algorithms simulated annealing*, pp. 42–57, 1987.

[36] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, “Evolution algorithms in combinatorial optimization,” *Parallel Comput.*, vol. 7, no. 1, pp. 65–85, 1988.

[37] P. Moscato, “On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms,” *Caltech Concurr. Comput. program, C3P Rep.*, vol. 826, p. 1989, 1989.

[38] Y. Chakrapani and K. S. Rajan, “Hybrid Genetic-Simulated Annealing Approach for Fractal Image Compression,” vol. 2, no. c, pp. 587–592, 2008.

[39] F. K. Mohamed and B. Aoued, “Optimization of fractal image compression based on genetic algorithms,” in *2nd International Symposium on Communications, Control and Signal Processing, Marrakesh, Marocco*, 2006.

[40] W. Xing-Yuan, L. Fan-Ping, and W. Shu-Guo, “Fractal image compression based on spatial correlation and hybrid genetic algorithm,” *J. Vis. Commun. Image Represent.*, vol. 20, no. 8, pp. 505–510, 2009.

[41] S. K. Roy, S. Kumar, B. Chanda, B. B. Chaudhuri, and S. Banerjee, “Fractal image compression using upper bound on scaling parameter,” *Chaos, Solitons & Fractals*, vol. 106, pp. 16–22, 2018.

[42] S. S. Al-Bundi, N. M. Al-Saidi, and N. J. Al-Jawari, “Crowding optimization method to improve fractal image compressions based iterated function systems,” *Int. J. Adv. Comput. Sci. Appl*, vol. 7, no. 7, pp. 392–401, 2016.

[43] S. Das and D. Ghoshal, “Hybrid Fractal Image Compression Based on Graph Theory and Equilateral Triangle Segmentation,” *Int. J. Appl. Eng. Res.*, vol. 11, no. 6, pp. 4467–4477, 2016.

[44] B. M. Ismail, B. E. Reddy, and T. B. Reddy, “Cuckoo inspired fast search algorithm for fractal image encoding,” *J. King Saud Univ. Inf. Sci.*, vol. 30, no. 4, pp. 462–469, 2018.

[45] S. Liu, Z. Zhang, L. Qi, and M. Ma, “A fractal image encoding method based on statistical loss used in agricultural image compression,” *Multimed. Tools Appl.*, vol. 75, no. 23, pp. 15525–15536, 2016.

[46] W. Xing-Yuan, Z. Dou-Dou, and W. Na, “Fractal image coding algorithm using particle swarm optimisation and hybrid quadtree partition scheme,” *IET Image Process.*, vol. 9, no. 2, pp. 153–161, 2014.

[47] G. Vahdati, H. Khodadadi, M. Yaghoobi, and M.-R. Akbarzadeh-T, “Fractal image compression based on spatial correlation and hybrid particle swarm optimization with genetic algorithm,” in *2010 2nd International Conference on Software Technology and*

Engineering, 2010, vol. 2, pp. V2-185.

[48] Y.-L. Lin and M.-S. Wu, "An edge property-based neighborhood region search strategy for fractal image compression," *Comput. Math. with Appl.*, vol. 62, no. 1, pp. 310–318, 2011.

[49] P. C. Cosman, R. M. Gray, and R. A. Olshen, "Evaluating quality of compressed medical images: SNR, subjective rating, and diagnostic accuracy," *Proc. IEEE*, vol. 82, no. 6, pp. 919–932, 1994.

[50] R. Menassel, I. Gaba, K. Titi, I. Gaba, and K. Titi, "Introducing BAT inspired algorithm to improve fractal image compression," *Int. J. Comput. Appl.*, vol. 0, no. 0, pp. 1–8, 2019.

[51] R. Menassel, B. Nini, and T. Mekhaznia, "An improved fractal image compression using wolf pack algorithm," *J. Exp. Theor. Artif. Intell.*, no. November 2018, pp. 1–11, 2017.

[52] Kalunge VV,VarunakshiB.Time optimization of fractal image compression by using genetic algorithm.Int J Eng Res Technol(IJERT).2012;1(10):1–5.

[53] WangX-Y,ZhangD-D.Discretewavelettransform-basedsimplerangeclassification strategies for fractal image coding. Nonlinear Dyn. 2014;75(3): 439–448.

[54] WangX-Y,ZhangD-D,WeiN.Fracta limage coding algorithm using particle swarm optimization and hybrid Quadtree partition scheme .IET Image Proc. 2015;9(2):153–161.

[55] Muruganandhama A, Banu W. Adaptive fractal image compression using PSO. Procedia Comput Sci. 2010;2:338–344. DOI:10.1016/j.procs.2010. 11.044

[56] ChakrapaniY,SoundararajanK.Implementation of fractal image compression employing particle swarm optimization .ISSN1746-7233,England,UK World JModel Simul .2010,6,1:40-46.

[57] Mitra SK, Murthy C a, Kundu MK. Technique for fractal image compression using genetic algorithm. IEEE Trans Image Process. 1998;7(4): 586–593