الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mohamed Khider - Biskra
Faculté des sciences et de la technologie
Département : Génie électrique
Ref: .....

جامعة محمد خيضر بسكرة
كلية العلوم و التكنولوجيا
قسم : الهندسة الكهربائية
المرجع : .....

Thèse présentée en vue de l'obtention
du diplôme de

Doctorat LMD en Génie Electrique

Spécialité : Electronique

Option : Signaux et Communications

Estimation de mouvement par les techniques métaheuristiques

Présentée par:

Abir BETKA

Soutenue publiquement le : 08/07/2019

Devant le jury composé de :

BAARIR Zine-Eddine	Professeur	Président	Université de Biskra
TERKI Nadjiba	Professeur	Directrice	Université de Biskra
TOUMI Abida	MCA	Co-directrice	Université de Biskra
RAMDANI Messaoud	Professeur	Examinateur	Université d'Annaba
LACHOURI Abderezak	Professeur	Examinateur	Université de Skikda
SBAA Salim	Professeur	Examinateur	Université de Biskra

Dédicace

Je dédie cette thèse à ..

Ma chère mère

Mon cher père

Ma chère grand-mère et mes chères tantes

Ma chère sœur Safynez et ses enfants

Mon cher frère Brahim et mes chères sœurs Manel, Fatima et Nour El-houda.

Toute la famille

Mes chères amies

Tous mes enseignants

Tous ceux que j'estime et qui me sont chers.

Remerciements

Je tiens premièrement à prosterner remerciant **ALLAH** le tout puissant de m'avoir donné le courage et la patience pour terminer cette thèse.

Je remercie ma mère et toute ma famille pour leur encouragement, leur soutien et leur sacrifice durant toutes mes années d'étude.

Je remercie la directrice et la co-directrice de cette thèse les professeurs TERKI Nadjiba et TOUMI Abida pour m'avoir encadré, guidé et encouragé pour réaliser cette thèse. Je les remercie également pour la patience et pour la confiance qu'elles m'ont toujours accordée

Je remercie le président du jury Pr. BAARIR Zine-Eddine ainsi que les membres Pr. RAMDANI Messaoud, Pr. LACHOURI Abderezak et Pr. SBAA Salim, qui ont accepté de juger mon travail.

Je remercie mes co-auteurs Noureddine Samia, HAMIANE Madina, OUAHABI Abdeljalil, DAHMANI Habiba et OURCHANI Amina, pour leur précieuse aide.

Je remercie aussi tous mes amis et mes camarades qui m'ont beaucoup soutenu conseillé et aidé.

Je remercie également tous mes enseignants durant toutes mes études, ainsi que toutes les personnes qui ont participé de prés ou de loin à la réalisation de cette thèse.

Résumé

L'estimation de mouvement est un processus qui consiste à estimer, à partir d'une séquence d'images, le mouvement apparent des objets composant une scène tridimensionnelle. La méthode de mise en correspondance de blocs : *block matching* (BM) est la méthode d'estimation de mouvement la plus utilisée. L'aspect important de cette méthode est l'utilisation des stratégies de recherche intelligentes afin d'obtenir une précision d'estimation de mouvement élevée avec une complexité de calcul réduite. Dans cette thèse, nous nous intéressons aux techniques de BM basées sur les métaheuristiques, deux nouveaux algorithmes sont proposés. Dans la première partie de cette thèse, nous proposons un algorithme de BM basé sur la recherche fractale stochastique. Les résultats de simulation obtenus montrent la supériorité de l'algorithme proposé par rapport aux autres techniques BM en termes de précision d'estimation de mouvement et complexité de calcul. Dans une deuxième partie, nous présentons la technique métaheuristique que nous avons développée, nommée optimisation à base de LBP : *local binary pattern optimizer* (LBPO), elle est inspirée par le concept de base du descripteur LBP. Nous validons la méthode LBPO avec des fonctions de test connues, puis nous appliquons cette méthode au problème de BM. Les résultats expérimentaux montrent l'efficacité de la méthode proposée.

Mots clés: Estimation de mouvement, Séquence d'images, Optimisation, Métaheuristique.

Abstract

Motion estimation is a process that consists on estimating, from a sequence of images, the apparent motion of the objects composing a three-dimensional scene. The block matching (BM) method is the most widely used motion estimation method. The important aspect of this method is the use of intelligent search strategies to obtain high motion estimation accuracy with reduced computational complexity. In this thesis, we are interested in BM techniques based on metaheuristics. Two new BM algorithms based on metheuristic techniques are proposed. In the first part of this thesis, we propose a BM algorithm based on the stochastic fractal search. The obtained simulation results show the superiority of the proposed algorithm over other BM techniques in terms of motion estimation precision and computational complexity. In the second part, we present the metheuristic technique that we have developed, called local binary pattern optimizer (LBPO), it is inspired by the basic concept of the LBP descriptor. We validate the LBPO method with test functions, then we apply this method to the BM problem. The experimental results show the effectiveness of the proposed method.

Keywords: Motion estimation, sequence of images, Optimization, Metaheuristics.

ملخـــــص

تقدير الحركة هي عملية لتقدير الحركة الظاهرة للأجسام التي تشكل مشهد ثلاثي الأبعاد وذلك انطلاقا من سلاسل الصور. طريقة تحديد الكـــتل المتوافقة هي الطريقة الأكثر استعمـــالا لتقدير الحركة. الجـــانب المهم في هذه الطريقة هو استعمـــال استراتيجيات بحث ذكية من أجل الحصول على دقة عالية في تقدير الحركة مع تعقيد حسابي منخفض. في هذه الأطروحة نحن مهتمون بتقنيات تحديد الكتل المتوافقة المعتمدة على الطرق الاستكشـافية. نقترح خوارزميتين جديدتين لتحديد الكتـــل المتوافقة معتمدين على الطرق الاستكشافية. في الجزء الأول من هذه الأطروحة نقترح خوارزميه لتحديد الكتـــل المتوافقة معتمدة على البحث العشوائي المنكسر. تظهر نتائـــج المحاكاة التي تم الحصـــول عليها تفوق الخوارزميه المقترحة مقارنـــة بتقنيات اخرى من حيث دقه تقدير الحركة والتعقيد الحسابي. في الجزء الثاني نقدم التقنية الاستكشافية التي طورناها والتي سميت التحسين علي أسـاس النموذج الثنائـي المحلـي وهي مستوحـاة من المفهوم الأساسـي لنموذج الثنائي المحلـــي. نقوم بالتحقق من مردوديـــة الطريقة باستعمال دوال اختبـار معروفـــة و من ثم نطبق هذه الطريقة لتحديـد الكتل المتوافقة. تبين النتائج التجريبيـة فعاليـة الطريقـة المقترحة.

الكلمات الدلائلية : تقدير الحركة، سلاسل الصور، تحسين، الاستكشافية.

Production scientifique

Publications internationales

BETKA Abir, TERKI Nadjiba, TOUMI Abida, et al. “A new block matching algorithm based on stochastic fractal search”. *Applied Intelligence*, 2019, vol. 49, no 3, p. 1146-1160.

ABDESSELAM, Salim, **BETKA Abir**, TOUMI Abida, Zine-eddine BAARIR. “Application of Stand-PSO Technique for Optimization Cameras’ 2D Dispositions in a MoCap system”. *Journal of Applied Computer Science and Mathematics*, 2016, no 21.

Communications internationales

Abir BETKA, Nadjiba TERKI, Abida TOUMI, Habiba DAHMANI, “An Enhanced GWO Algorithm for Solving Block Matching Problem”. *The 2nd International Conference on Electrical Engineering, ICEEB’2018*.

Abida TOUMI, Fatiha HANI, **Abir BETKA** and Salim SBAA. “Knapsack Problem Resolution Based on Optics Inspired Optimization Algorithm”. *The fifth international conference on image and signal processing and their applications, ISPA’17 Mostaganem*.

Imane KHATA, Abida TOUMI, **Abir BETKA** and Salim SBAA. “Mobile Network Design Problem Resolution Based on Colliding Bodies Optimization Algorithm”. *The fifth international conference on image and signal processing and their applications, ISPA’17 Mostaganem*.

Amina OURCHANI, Zine-eddine BAARIR, Abdelmalik TALEB-AHMED and **Abir BETKA**. “Foreground detection using Block-matching search-LBP algorithm for dynamic background video”. *The fifth international conference on image and signal processing and their applications, ISPA’17 Mostaganem*.

Amina OURCHANI, Zine-eddine BAARIR, Abdelmalik TALEB-AHMED and **Abir BETKA**. “Object detection from dynamic background”. I*nternational conference on auto-*

matic control, telecommunications and signals, (ICATS'17), annaba, ALGERIA.

Samia Nouredine, Abida Toumi, **Abir Betka**. "Symbiotic Approach for Datamining". *The 4th International Conference of Computing for Engineering and Sciences*, "ACM-ICCES - 2018" 14-16 July 2018 - Kuala Lumpur, Malaysia.

Abida TOUMI, **Abir BETKA**, Salim SBAA, Nadjiba TERKI. "WPBO: A New Metaheuristic Technique Inspired from Wolf Pack Behaviour". *The META'2018 International conference on Metaheuristics and Nature Inspired Computing.*

Table des matières

Table des figures

Liste des Algorithmes

Liste des tableaux

Abréviations

EM	Estimation de Mouvement
BM	Block Matching
FSA	Full Search Algorithm
SFS	Stochastic Fractal Search
LBPO	Local Binary Pattern Optimizer
2D	Deux Dimensions
3D	Trois Dimensions
EFO	Équation de Flot Optique
FFD	Free-Form Deformations
NCC	Corrélation Croisée Normalisée
HSV	Hue Saturation Value
RVB	Rouge Vert Bleu
SAD	Somme des Différences Absolues
SSD	Somme Square Difference
CC	Cœfficient d'inter-Corrélation
PSNR	Peak Signal to Noise Ratio
MSE	Mean Square Error
MPEG	Moving Picture Experts Group
TSS	Three Step Search
NTSS	New Three Step Search
4SS	Four Step Search
OSA	Orthogonal Search Algorithm
CSA	Cross Search Algorithm
DS	Diamond Search
CZS	Circular Zonal Search
OTS	One-at-a-Time Search
2DLOG	Two Dimensional LOGarithmic
ARPS	Adaptive Rood Pattern Search
BBGD	Block-Based Gradient Descent search
EA	Evolutionary Algorithms
SI	Swarm Intelligence

NFL	No Free Lunch
PSO	Particle Swarm Optimization
DE	Differential Evolution
ABC	Artificial Bee Colony
HS	Harmony Search
SEA	Successive Elimination Algorithm
MPBM	Mean Prediction Block Matching
SDDS	Star Diamante Diamante Search
BH	Black Hole
SCA	Sine-Cos Algorithm
GWO	Grey Wolf Algorithm
SSA	Salp Swarm Optimization
MS	Moth Search
PLA	Pursuit Learning Automata
TPLA	Team Pursuit Learning Automata
LBP	Local Binary Pattern
CA	Cultural Algorithm

Introduction Générale

La séquence vidéo est une source d'informations et un support de communication très important ; elle peut présenter l'idée mieux qu'une description écrite. Elle est même parmi les types de données les plus vues, partagés et consommés dans le monde ; des milliards de vidéos sont vues chaque jour sur les différentes plateformes. Les séquences vidéos peuvent contenir des informations très importantes, il est alors très intéressant de développer des techniques d'analyse des vidéos pour extraire ces informations. Parmi ces dernières, on trouve les informations liées au mouvement, qui jouent un rôle essentiel pour des applications diverses comme la compression vidéo, la robotique, la surveillance, la météorologie, l'océanographie, les applications médicales, militaires...

L'estimation de mouvement (EM) à partir d'une séquence vidéo est un processus par lequel les informations liées aux mouvements sont extraites. Il consiste à quantifier les déplacements des pixels entre les images. Depuis 35 ans, de nombreuse techniques d'EM ont été proposées et améliorées progressivement. Nous classifions les techniques d'EM en deux grandes catégories. Les méthodes de flot optique (à base de pixel) qui calculent un vecteur de mouvement pour chaque pixel dans l'image [1]-[6], et les méthodes de mise en correspondance de blocs (BM : block matching) qui divisent l'image en blocs de pixels et calculent ensuite un vecteur de mouvement pour chaque bloc [7], [9]-[13]. Les méthodes de flot optique procurent un flot optique dense, en calculant un vecteur de mouvement pour chaque pixel. Cependant, ces méthodes demandent un temps de calcul excessif, et elles donnent des résultats satisfaisants seulement si le mouvement dans la séquence d'images est faible.

Les méthodes de mise en correspondance de blocs ou block matching (BM) sont les méthodes d'EM les plus utilisées grâce à leur implémentation simple et efficace. Le principe général de ces méthodes est que chaque deux images successives dans la séquence vidéo (courante et référence) sont divisées en blocs. Ensuite, pour chaque bloc dans l'image courante, l'algorithme cherche le bloc le plus similaire dans l'image de référence en minimisant une erreur de dis-similarité entre les illuminations correspondantes aux pixels respectifs des deux blocs. Le vecteur de mouvement est la différence entre les positions du bloc courant et du bloc le plus similaire dans l'image de référence.

La première méthode de BM adopte une stratégie de recherche exhaustive (FSA : full search algorithm) dans l'image de référence en évaluant tous les blocs existants dans la fenêtre de recherche [8]. Avec la stratégie de recherche exhaustive, le bloc le plus similaire est trouvé, et par conséquence, la précision d'estimation de mouvement est élevée. Cependant, elle est très lourde en temps de calcul, ce qui rebute certains utilisateurs. Pour remédier à ce problème, plusieurs méthodes de BM avec des stratégies de recherche intelligentes ont été proposées pour réduire la complexité de calcul de FSA et assurer au même temps que la dégradation introduite au niveau de la précision, ne sera pas gênante. Parmi eux, on trouve les méthodes de BM orientées comme TSS [9], OSA [14] et DS [15]. Malgré que ces méthodes réduisent effectivement la complexité de calcul, mais elles peuvent tomber sur des solutions sous-optimales (minima locaux). Il y a aussi les méthodes BM avec une recherche exhaustive rapide telles que [16, 17], et récemment, le problème de BM a été traité comme un problème d'optimisation, et différentes techniques métaheuristiques ont été utilisées pour le résoudre comme PSO-BM [11, 18], ABC-BM [19], DE-BM [20] et HS-BM [12].

Les métaheuristiques sont des techniques d'optimisation approximatives, permettent l'obtention des résultats d'optimisation satisfaisants dans un temps raisonnable. Elles se basent sur la minimisation (ou maximisation) d'une fonction objectif, et sur des mécanismes aléatoires pour explorer l'espace de recherche, afin de déterminer ou d'approcher un optimum global. Les métaheuristiques ont été appliquées avec succès dans plusieurs domaines de recherche comme les réseaux de télécommunications, les applications médicales, les systèmes électriques, les applications multimédias, en traitement d'image et notamment en estimation de mouvement [21].

Dans cette vision, nous avons décidé de travailler avec les méthodes BM basées sur les techniques métaheuristiques. Les techniques métaheuristiques utilisées dans ce contexte souffrent d'un nombre de problèmes, pour cela, nous visons dans cette thèse à proposer de nouvelles méthodes BM basées sur des métaheuristiques capables d'obtenir des résultats d'estimation de mouvement satisfaisants en termes de précision et complexité de calcul. Nos principales contributions sont :

- Dans la première contribution, nous proposons un nouvel algorithme de BM basé sur la technique métaheuristique de recherche fractale stochastique (SFS : stochastic fractal search). Une implémentation parallèle est utilisée pour calculer les déplacements de tous les blocs simultanément. Pour améliorer davantage les performances, de nouvelles

idées concernant l'initialisation, la fonction de fitness, la fenêtre de recherche et l'approximation de la fonction de fitness sont proposées.

- Dans la deuxième contribution, nous proposons une nouvelle technique métaheuristique nommée l'optimisation à base de LBP (LBPO : local binary pattern optimizer). Elle est inspirée du concept de base de l'opérateur LBP (local binary pattern). Un nouveau algorithme de BM basé sur LBPO est ensuite proposé.

La suite de cette thèse est organisée de la manière suivante :

Chapitre 1 : ce chapitre donne le principe général du problème de l'estimation de mouvement, ainsi que sa formulation mathématique et les difficultés soulevées lors de la résolution de ce problème. Nous présentons ensuite les deux grandes catégories des techniques d'estimation de mouvement en mettant l'accent sur la possibilité de résoudre ce problème avec les métaheuristiques.

Chapitre 2 : est consacré aux techniques métaheuristiques. Nous décrivons tout d'abord le problème d'optimisation, ensuite, nous nous concentrons principalement sur les techniques d'optimisation métaheuristiques en détaillant le principe général, classification de différentes techniques connues dans la littérature et les extensions possibles.

Chapitre 3 : présente notre première contribution, qui est l'utilisation de la technique métaheuristique SFS pour l'estimation de mouvement. Nous décrivons d'abord l'algorithme SFS, ensuite, nous présentons l'algorithme SFS-BM.

Chapitre 4 : dans ce chapitre, nous proposons notre nouvelle méthode métaheuristique LBPO. Après avoir détaillé son inspiration et son modèle mathématique, LBPO est évaluée sur des fonctions de test, ensuite, appliquée en BM.

Chapitre I : État de l'art sur l'estimation de mouvement

Sommaire

I.1 Introduction

L'estimation de mouvement (EM) est un processus par lequel les déplacements des pixels dans une séquence vidéo sont quantifiés. C'est un processus qui s'appuie sur la conservation temporelle d'intensité au cours du déplacement. L'information de mouvement extraite joue un rôle primordial dans des applications diverses dans le domaine de la vision par ordinateur comme la compression vidéo, la robotique, la surveillance, etc.
Lors de l'EM, on se trouve face à un problème inverse, mal posé et difficile à résoudre à cause de plusieurs raisons théoriques et pratiques comme la perte d'information lors de la projection, les restrictions de l'hypothèse de la conservation d'intensité, la présence de bruit dans les images, etc. Les techniques d'EM peuvent être regroupées en deux grandes catégories : les méthodes de flot optique (à base de pixel) et les méthodes de mise en correspondance de blocs (à base de bloc).
Dans ce chapitre, nous expliquerons le principe de l'EM, ensuite, nous exposerons les grandes catégories des techniques d'EM en mettant l'accent sur la possibilité à considérer l'EM comme un problème d'optimisation, et ce dernier peut être résolu par les techniques métaheuristiques.

I.2 Le principe de l'estimation de mouvement

La séquence vidéo est une suite d'images fixes qui défilent à une certaine cadence. Cette cadence est de l'ordre de 25 images par seconde que l'œil humain peut la considérer comme une image animée (voir figure. I.1). Généralement une image donnée de la séquence vidéo et les images qui l'entourent (précédent ou succédant) présentent une forte similarité, elles sont composées d'un même ensemble d'objets qui changent de position en chaque image [22].

I.2.1 Le mouvement dans la séquence vidéo

Généralement, l'image est le résultat de la projection d'une scène réelle tridimensionnelle sur le plan de l'image bidimensionnelle (voir figure. I.2). Le processus d'EM cherche donc à recouvrir la projection des différents mouvements tridimensionnels de la scène sur le plan image. De façon formelle, il consiste à estimer à partir d'une séquence d'images le mouvement apparent des objets composant une scène tridimensionnelle. La caméra ou les objets peuvent être selon les cas mobiles ou immobiles [23].

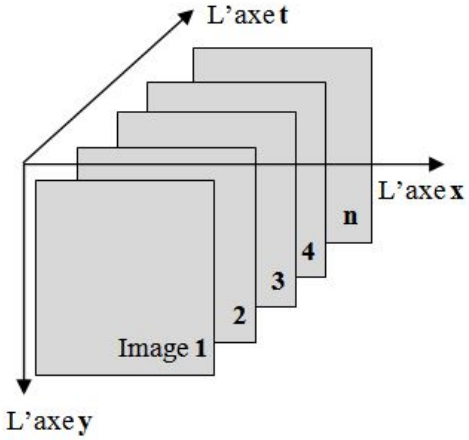


FIGURE I.1. : La séquence vidéo

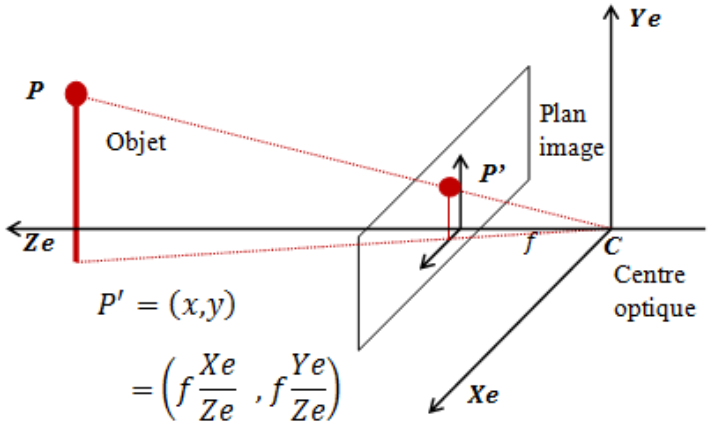


FIGURE I.2. : Projection 3D en 2D

L'EM est un problème extrêmement difficile à résoudre [23]. En effet, d'une part la projection 3D en 2D cause une perte d'information, et d'autre part, les variations d'intensité lumineuse sont les seules données disponibles. Donc, à partir des images 2D et en utilisant les variations d'intensité lumineuse, nous pouvons accéder au mouvement apparent, appelé aussi flot optique, traduisant de manière plus ou moins fidèle le mouvement réel (voir figure. I.3) [23, 24, 25, 26] [244].

FIGURE I.3. : Exemples de flot optique

I.2.2 Formulation mathématique

Soient I_1 et I_2 deux images successives dans une séquence vidéo. L'objectif est d'estimer les valeurs de flot optique $\omega : \Omega \rightarrow R^2$ entre ces deux images. Pour chaque pixel dans l'image $\mathbf{x} = (x,y)^\top \subset R^2$, le flot optique $\omega(\mathbf{x})$ est composé de deux valeurs selon les composantes x et y, $\omega(\mathbf{x}) = (u(\mathbf{x}), v(\mathbf{x}))^\top$.
Pour déterminer le flot optique à partir des intensités lumineuses, les méthodes d'EM commencent par choisir une hypothèse simplificatrice sur la variation d'intensité dans la séquence considérée. Cette hypothèse suppose que l'intensité lumineuse ou la couleur apparente du pixel $\mathbf{x}$, qui se déplace d'une image à une autre reste constante, c'est-à-dire l'intensité est indépendante du temps t, ce qui se traduit par l'équation suivante :

$$I_1(\mathbf{x}) = I_2(\mathbf{x} + \omega(\mathbf{x})) = constante \tag{I.1}$$

En supposant que les déplacements des pixels sont petits, le développement de Taylor du terme au milieu dans l'équation.(I.1) est comme suit :

$$I_2(\mathbf{x} + \omega(\mathbf{x})) = I_2(\mathbf{x}) + I_x.u(\mathbf{x}) + I_y.v(\mathbf{x}) \tag{I.2}$$

Où $I_x = \frac{\partial I_2}{\partial x}$ et $I_y = \frac{\partial I_2}{\partial y}$, avec $I_t = I_2 - I_1$ on obtient :

$$I_t + I_x.u + I_y.v = 0 \tag{I.3}$$

Ou encore

$$I_t + \nabla I.\omega = 0 \tag{I.4}$$

Avec $\nabla I = (\frac{\partial I_2}{\partial x}, \frac{\partial I_2}{\partial y})^\top$
La relation I.3 est appelée l'équation de flot optique. [27, 28, 29, 30].

L'équation de flot optique (EFO) décrit une droite dans l'espace de vitesses (u, v) comme illustrée sur la figure I.4.

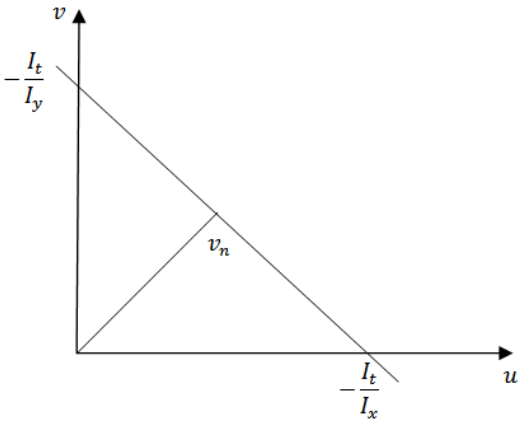


FIGURE I.4. : Illustration géométrique de l'équation de flot optique

Il faut noter que l'équation de flot optique est une seule contrainte à deux inconnues (u, v) pour chaque pixel. Il s'agit donc d'un problème inverse mal posé [23]. À partir de cette équation, il est possible d'estimer que la composante normale v_n de vecteur vitesse orientée dans la direction du gradient spatial d'intensité [27][23].

$$v_n = -\frac{I_t}{\| \nabla I \|} = -\frac{I_t}{\sqrt{{I_x}^2 + {I_y}^2}} \tag{I.5}$$

La résolution des problèmes inverses et mal posés, analytiquement, est extrêmement difficile, il est nécessaire d'ajouter une hypothèse supplémentaire pour réduire l'espace des possibilités et aboutir à une solution unique [27, 23, 24].

I.2.3 Les difficultés de l'estimation de mouvement

Malgré les progrès significatifs connus dans le domaine d'EM, mais une estimation de mouvement très précise reste difficile à obtenir à cause de nombreuses raisons théoriques et pratiques [22, 25, 31, 32, 33] :

- Le passage du 3D au 2D provoque une perte de certaines informations importantes de la scène. Des points différents de l'espace peuvent être projetés sur le même point de l'image, une droite peut être représentée par un point et un plan peut devenir une droite, ce qui pose entre autres des problèmes de mise en correspondance.

- L'hypothèse de la conservation de la luminance d'un pixel au cours de déplacement combine effectivement plusieurs d'autres hypothèses concernant la réflectivité de la scène (la scène est lambertienne), l'illumination de la scène (illumination uniforme) et la formation de l'image au niveau de la caméra (pas de vignettage).

- Le problème d'occlusion se manifeste lors du recouvrement d'une surface par une autre, dû au mouvement de la caméra ou des objets dans la scène, certains points sont déplacés à des positions où ils sont devenus invisibles (voir figure. I.5).

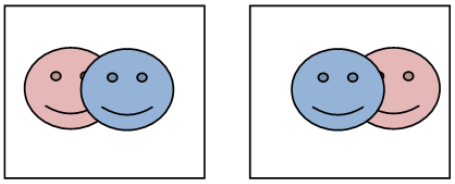


FIGURE I.5. : Problème d'occlusion

- Le problème de région transparente : lorsque l'objet mobile est transparent, l'hypothèse de la conservation d'intensité n'est pas valide.

- Le problème de région homogène : un exemple sur ce problème est celui d'une sphère homogène tournant sur elle-même. Malgré qu'un mouvement réel est existant, mais aucune variation d'intensité n'est induite, donc le flot optique est nul.

- Le problème d'ouverture est une reformulation du fait que la solution du problème d'EM n'est pas unique. Dans la figure.I.6, un objet rectangulaire qui se déplace avec un mouvement uniforme, trois cas pratiques s'apparaissent lors de l'EM.

    1. Dans la zone 1 : la zone est homogène donc le gradient d'intensité est nul. Il n'est pas possible d'en déduire le véritable mouvement

2. Dans la zone 2 : le gradient d'intensité est orienté vers une seule direction. Le mouvement estimé est normal au contour.

3. Dans la zone 3 : plusieurs gradients d'intensité sont non nuls. Le mouvement réel est estimé correctement [25].

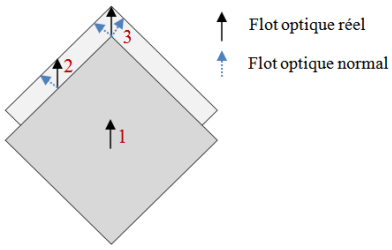


FIGURE I.6. : Problème d'ouverture

- La présence du bruit : plusieurs sources contribuent à la présence du bruit dans la séquence vidéo tel que le bruit du capteur, le bruit dû aux erreurs de transmission de données, le bruit de quantification, etc.

I.2.4 Domaines d'applications

L'estimation de mouvement est une étape essentielle pour plusieurs applications dans le domaine de la vision par ordinateur. En compression de vidéo [34], l'EM exploite la redondance temporelle entre les images pour un gain de compression significatif ; cela est possible en représentant l'image t à partir de l'image $t-1$ de la séquence et en transmettant uniquement les vecteurs de mouvement des blocs et l'erreur résiduelle entre les deux images [27]. L'EM revêt ainsi une importance primordiale en robotique pour le guidage automatique des engins, la conduite autonome des voitures, le contrôle des actions des robots, etc [35, 36, 37]. La surveillance automatique est une application intéressante de l'EM ; en particulier pour l'identification des gestes et expressions faciales, l'analyse de mouvement de foule et comportement de piétons [38, 39, 40]. Dans le domaine médical, l'EM peut être utilisé pour l'analyse de mouvement du cœur, l'étude des écoulements bio-fluides et les déformations des organes [41, 42, 43]. En traitement d'image, les vecteurs de mouvement obtenus peuvent être utilisés à des fins diverses comme le filtrage, la segmentation, la restauration d'image, la reconstruction 3D, etc [44, 45, 46]. Dans les dernières années, on trouve aussi l'EM dans des applications militaires pour la poursuite des cibles, dans la météorologie pour l'étude de champs de vents,

et dans l'océanographie pour l'étude des courants marins [47, 48, 49], etc.

I.3 Les méthodes d'estimation de mouvement

Dans cette section, nous présentons les méthodes d'EM. Nous n'allons pas citer toutes les méthodes existantes, mais essayer de proposer une classification des principales méthodes développées depuis trois décennies. Nous nous basons sur des études d'état de l'art menées sur les techniques d'EM comme [26], [50]-[53].
Les techniques d'EM peuvent être regroupées en deux grandes catégories : les méthodes de flot optique qui calculent un vecteur de mouvement pour chaque pixel dans l'image et les méthodes de mise en correspondance de blocs qui divisent l'image en blocs de pixels et calculent ensuite un vecteur de mouvement pour chaque bloc.

I.3.1 Les méthodes de flot optique (à base de pixel)

Les méthodes de flot optique estiment les valeurs de flot optique pour chaque pixel dans l'image, elles se basent sur la minimisation d'une fonctionnelle composée de l'équation du flot optique et une contrainte supplémentaire de lissage ou régularisation pour particulariser les solutions.
Le rôle de la deuxième contrainte est de lever l'indétermination de l'estimation, par la prise en compte d'un voisinage spatial en propageant l'information vers les zones trop uniformes. Elle consiste à imposer à la solution recherchée une certaine "forme" mathématique en introduisant une dépendance spatiale mutuelle entre les vecteurs de vitesse des pixels dans l'image. Cette dépendance peut, suivant les cas, s'exprimer de façon locale ou globale. On peut donc classifier les méthodes de flot optique en deux catégories : locale et globale [22, 23]. Le tableau. I.1 représente quelques méthodes de flot optique.

I.3.1.1 Les méthodes locales

Les méthodes locales consistent à minimiser l'hypothèse supplémentaire sur un domaine plus petit que l'image entière [22]. La méthode locale la plus célèbre est celle de **Lucas et Kanade** [2], elle estime les valeurs de flot optique pour chaque pixel, cela se fait en supposant que le flot optique est localement constant sur un voisinage plus petit que l'image entière.

TABLE I.1. : Quelques méthodes flot optique

Catégorie		Méthode	Référence
Locale		La méthode de Lucas et Kanade	Lucas et Kanade (1981) [2]
		Modèle polynomiale	Black, et al. (1996) [54]
		Modèle obtenu par apprentissage	Black, et al. (1997) [55]
		Modèle de forme libre	Karantzalos, et al. (2005) [56]
Globale		Norme L^2	Horn et Schunck (1981) [1]
		Norme L^1	Brox, et al. (2004) [3]
		La fonction de Tukey	Odobez, et al. (1995) [57]
		La norme Lorentzian	Black, et al. (1996) [54]
	Le terme d'attache aux données	Conservation du gradient de l'image	Uras, et al. (1988) [83]
		Conservation du laplacien de l'image	Papenberg, et al. (2006) [58]
		Utilisation de la texture	Wedel, et al. (2009) [59]
		Investigation dans l'espace de couleur	Golland, et al. (1997) [60]
		Combinaison de plusieurs filtres	Weber, et al (1995) [61]
		Combinaison de différentes terme de conservation	Hyun, et al. (2013) [62]
	Le terme de Régularisation	Norme L^2	Horn et Schunck (1981) [1]
		Norme L^1	Shulman, et al. (1989) [63]
		Les dérivés seconde	Trobin, et al. (2008) [64]
		Champ aléatoire de Markov	Heitz, et al (1993) [65]
		La variation totale	Wedel, et al. (2009) [59]
		Régularisation non-locale	Krähenbühl, et al. (2012) [4]
		Régularisation temporelle	Chin, et al. (1994) [66]
Récentes		Gérer les occlusions	Kolmogorov,et al. (2001) [67]
			Ayvaci, et al. (2012) [68]
			Leordeanu, et al. (2013) [69]
		Larges déplacements	Hu, et al. (2017) [70]
			Fan, et al. (2018) [5]
			Sun, et al. (2018) [71]
		Autres	Schuster, et al. (2018) [72]
			Trinh, et al. (2018) [73]
			Zarei-Jalalabadi, et al. (2018) [74]
			TU, et al. (2019) [75]
			Khalid, et al. (2019) [6]

Cette méthode consiste alors à minimiser l'équation du flot optique localement en utilisant la méthode des moindres carrés pondérés dans une fenêtre $W(x,y) \in \Omega$ [76, 77].

$$E(\omega(\mathbf{x})) = \sum_{W} W(x,y)(I_t + \nabla I \cdot \omega)^2 \tag{I.6}$$

Où $W(x,y)$ est une fenêtre de pondération spatiale. Généralement, c'est une fenêtre gaussienne ; ses valeurs décroissent avec l'augmentation de la distance entre le pixel central et le pixel voisin.
Le minimum de ce système correspond aux points critiques, où les dérivées selon u et v sont nulles.

$$\begin{array}{rcl} \frac{\partial(E(\omega(\mathbf{x})))}{\partial u} & = & \sum_{W} W(x,y)(uI_x^2 + vI_xI_y + I_xI_t) = 0 \\ \frac{\partial(E(\omega(\mathbf{x})))}{\partial v} & = & \sum_{W} W(x,y)(vI_y^2 + uI_xI_y + I_yI_t) = 0 \end{array} \tag{I.7}$$

Ces deux équations peuvent être réécrites sous la forme matricielle suivante :

$$\begin{bmatrix} \sum WI_x^2 & \sum WI_xI_y \\ \sum WI_xI_y & \sum WI_y^2 \end{bmatrix} . \quad \omega(\mathbf{x}) = \quad - \begin{bmatrix} \sum WI_xI_t \\ \sum WI_yI_t \end{bmatrix} \tag{I.8}$$

Avec M et b sont :

$$M = \begin{bmatrix} \sum WI_x^2 & \sum WI_xI_y \\ \sum WI_xI_y & \sum WI_y^2 \end{bmatrix}$$

$$b = - \begin{bmatrix} \sum WI_xI_t \\ \sum WI_yI_t \end{bmatrix}$$

L'estimateur au sens moindre carré nous donne :

$$\omega(\mathbf{x}) = M^{-1}.b \tag{I.9}$$

Parmi les avantages de la méthode de Lucas et Kanade, on peut citer :

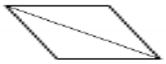
- Parallélisable, c'est-à-dire les calculs peuvent être effectué d'une façon parallèle ;
- Les résultats sont moins sensibles au bruit ;
- Elle permet le calcul des mouvements locaux ;

- Implémentation facile.

Malheureusement, cette méthode génère également certains inconvénients au niveau des mouvements complexes où le modèle translationnel de mouvement sur lequel la méthode de Lucas et Kanade se base, n'est pas toujours valide [22, 78]. Pour remédier à ce problème, des modèles de mouvement paramétriques ont été proposés pour inclure des mouvements plus complexes. Le calcul des paramètres nécessite d'abord l'estimation du déplacement d'un certain nombre de points de contrôle, ensuite, à partir des déplacements estimés, les paramètres sont calculés en utilisant la méthode des moindres carrés [24, 26, 79].

Il est difficile de repérer tous les mouvements complexes avec un seul modèle, c'est pour cela que plusieurs modèles de mouvement ont été proposés [54]. Le choix du modèle approprié dépend de l'application et des informations a priori concernant le mouvement dans la séquence vidéo. Le tableau. I.2 donne quelques modèles paramétriques de forme polynomiale, et un exemple d'une image de test pour chaque modèle [79].

TABLE I.2. : Quelques modèles paramétriques

Transformation	Modèle	Image de test
Euclidienne	$u(x,y) = cos(\alpha).x - sin(\alpha).y + d_u$ $v(x,y) = sin(\alpha).x - cos(\alpha).y + d_v$	
Affine	$u(x,y) = a_u.cos(\alpha).x - b_u.sin(\alpha).y + d_u$ $v(x,y) = a_v.sin(\alpha).x - b_v.cos(\alpha).y + d_v$	
Projective	$u(x,y) = (a_u.x - b_u.y + d_u)/(a.x - b.y + d)$ $v(x,y) = (a_v.x - b_v.y + d_v)/(a.x - b.y + d)$	
Bilinéaire	$u(x,y) = a_u.x - b_u.y + c_u.x.y + d_u$ $v(x,y) = a_v.x - b_v.y + c_v.x.y + d_v$	

Malgré que les modèles paramétriques peuvent être utilisés pour les mouvements complexes, l'inconvénient est qu'ils restent applicables seulement dans le cas de mouvement rigide. Le modèle de déformations de formes libres (FFD : free-form deformations) a été proposé pour repérer les mouvements non-rigides [80, 56, 81].

Le modèle de mouvement peut être aussi déterminé en utilisant des méthodes d'apprentissage. Souvent, la base d'apprentissage utilisée reflète une hypothèse sur la forme du flot optique existant et suit l'application désignée [26, 79, 55].

Il faut noter que lorsque les paramètres d'un modèle sont constants sur un domaine R, le champ de mouvement résultant est lisse et ne peut être valable que dans les régions de mouvement cohérent. Donc, la région R doit être assez grande pour permettre l'estimation de mouvement et aussi suffisamment petite pour maintenir la validité des paramètres. Plusieurs stratégies peuvent être utilisées pour la détermination de la région R :

- L'image entière, dans le cas d'un seul mouvement dans l'image ;
- Des blocs carrés d'une taille fixe, comme dans la méthode de Lucas et Kanade ;
- Des régions segmentées, les régions optimales peuvent être obtenues avec une segmentation de l'image à des régions cohérentes [26].

I.3.1.2 Les méthodes globales

Dans le cas où les images sont composées de régions homogènes ou un flot optique dense est nécessaire, les méthodes locales trouvent ses limites. Pour surmonter ces limitations, des méthodes globales ont été proposées. Ce type de méthodes consiste à minimiser, sur le domaine entier de l'image, une contrainte dite d'attache aux données basée sur l'équation du flot optique ainsi qu'une contrainte supplémentaire de lissage qui suppose que tous les pixels voisins ont un mouvement semblable [22, 24]. La première méthode globale proposée dans la littérature est celle de **Horn et Schunck** [1], elle consiste à trouver le champ de vitesse qui minimise la formule suivante :

$$E(\omega(\mathbf{x})) = \int (E_{Donnees}(\omega(\mathbf{x})) + \gamma . E_{Lissage}(\omega(\mathbf{x})))d\mathbf{x} \tag{I.10}$$

Où

$$E_{Donnees}(\omega(\mathbf{x})) = (I_t + I_x.u + I_y.v)^2 \tag{I.11}$$

$$E_{Lissage}(\omega(\mathbf{x})) = \| \nabla u \|^2 + \| \nabla v \|^2 = (\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2 \tag{I.12}$$

Avec γ qui est un paramètre positif qui permet d'ajuster l'influence de chaque terme. La minimisation de cette fonction peut-être effectuée en utilisant une méthode d'optimisation itérative comme la méthode de Gauss-Seidel.
La méthode de Horn et Schunck a des avantages et des inconvénients. Parmi ses principaux avantages :

- La simplicité d'implémentation ;
- Le flot optique généré est dense ;
- Elle donne de très bons résultats si les mouvements dans la séquence vidéo sont lisses.

Mais l'inconvénient majeur de cette méthode est qu'elle ne prend pas en considération la présence de discontinuité en mouvement et les petits mouvements sont ignorés à cause du lissage. Ce premier travail a été suivi d'un grand nombre de contributions qui ont proposé différentes variantes de terme de données et de lissage, afin d'améliorer les résultats [26].

Variantes du terme de données :

L'équation de flot optique suppose que l'intensité du pixel ne change pas avec le déplacement, mais la valeur d'intensité est tout à fait susceptible à de légers changements d'illumination, donc l'utilité de cette contrainte n'est pas toujours valide [26, 30, 3, 82].
Le tableau. I.3 présente quelque célèbres variantes de l'équation de flot optique. Au lieu de la conservation d'intensité, nous pouvons utiliser la conservation du gradient, de la matrice Hessienne ou l'opérateur laplacien, etc [83, 58].

Il est possible aussi d'utiliser d'autres mesures de similarité comme la corrélation croisée normalisée (NCC) qui est généralement invariante au changement d'illumination [84]. Le terme de données associé peut être écrit comme suit :

$$E_{Donnees} = 1 - NCC(\boldsymbol{\omega}(\mathbf{x})) \tag{I.13}$$

On trouve aussi des méthodes d'EM globales qui gardent l'équation de flot optique, et pour améliorer les résultats, elles effectuent des transformations sur les images, par l'exemple

TABLE I.3. : Quelques variantes de l'équation de flot optique

La conservation de	**La contrainte**
Gradient	$\nabla I_1(\mathbf{x}) = \nabla I_2(\mathbf{x}+\omega(\mathbf{x}))$
Hessienne	$H(I_1(\mathbf{x})) = H(I_2(\mathbf{x}+\omega(\mathbf{x})))$
Laplacien	$\Delta I_1(\mathbf{x}) = \Delta I_2(\mathbf{x}+\omega(\mathbf{x}))$
Norme de Gradient	$\parallel \nabla I_1(\mathbf{x}) \parallel = \parallel \nabla I_2(\mathbf{x}+\omega(\mathbf{x})) \parallel$
Norme de Hessienne	$\parallel H(I_1(\mathbf{x})) \parallel = \parallel H(I_2(\mathbf{x}+\omega(\mathbf{x}))) \parallel$
Déterminant de Hessienne	$det(H(I_1(\mathbf{x}))) = det(H(I_2(\mathbf{x}+\omega(\mathbf{x}))))$
Transformation de Census	$C(I_1(\mathbf{x})) = C(I_2(\mathbf{x}+\omega(\mathbf{x})))$

dans [3, 85] un filtre gaussien a été utilisé pour réduire le bruit dans les images. Dans [59] les auteurs ont décomposé l'image en deux parties structure et texture en utilisant la méthode de [86], ensuite, en se basant sur le principe que le changement d'illumination affecte la structure, ils ont utilisé pour l'estimation de mouvement seulement la partie texture de l'image. Les différents types d'espace de couleur ont été aussi exploités comme l'espace HSV (Hue saturation value) [85, 87, 88] et l'espace RVB normalisé [60], etc.

Dans le travail [89], les auteurs ont filtré le terme de données avec un filtre gaussien f comme suit :

$$E_{Donnees}(\omega(\mathbf{x})) = f(I_t + I_x.u + I_y.v) \tag{I.14}$$

Le filtrage du terme de données donne de très bons résultats surtout avec les séquences bruitées, mais il génère également une certaine dégradation au niveau de discontinuités et contours. Ce problème peut être résolu avec le remplacement du filtre gaussien par un filtre anisotropique [90].

La validité de chaque terme de données reste limitée à quelques situations, mais l'image est composée de plusieurs régions, chacune d'elles a ses propriétés et peut satisfaire à une contrainte et rejeter les autres. Pour cela, des termes de données plus complexes, combinant plusieurs contraintes ont été proposés. Cette combinaison peut être formulée comme la somme

de plusieurs termes pondérés ; l'équation.I.15 [65, 91, 62] :

$$E_{Donnees}(\omega(\mathbf{x})) = \sum_{p=1}^{P} \gamma_p(\mathbf{x}) E_p(\omega(\mathbf{x})) \tag{I.15}$$

Une autre façon de combinaison a été proposée dans [61], où K filtres différents sont appliqués aux images pour générer un système d'équation pour chaque pixel ; l'équation. I.16.

$$E_{Donnees}(\omega(\mathbf{x})) = \sum_{k=1}^{K} (f_k(I_1(\mathbf{x})) - f_k(I_2(\mathbf{x}+\omega(\mathbf{x})))) \tag{I.16}$$

Un modèle explicite du changement d'illumination peut être formulé comme suit :

$$g(\mathbf{x})I_1(\mathbf{x}) = I_2(\mathbf{x}+\omega(\mathbf{x})) + b(\mathbf{x}) \tag{I.17}$$

Le problème d'EM, avec ce modèle, est encore plus difficile à résoudre avec quatre inconnues et une seule équation, mais l'ajout des contraintes supplémentaires sur le changement d'illumination peut résoudre ce problème [30] [92].

Variantes de terme de lissage

Généralement, le lissage est modélisé par une pénalisation du module de gradient de flot optique en supposant que les pixels voisins du même objet ont des vitesses similaires. La méthode de Horn et Schunck utilise une pénalisation quadratique. Cette pénalisation facilite la résolution du problème, mais elle ne prend pas en considération la présence de discontinuités en flot optique, surtout quand des objets différents se déplacent dans des directions différentes. Pour cela, d'autres pénalisations robustes, qui augmentent moins rapidement que la pénalisation quadratique ont été proposées comme la norme L^1, L^1 modifié, Logarithmique et Lorentzian (voir tableau I.4 et figure. I.7) [26, 30, 53].

Une autre technique pour résoudre le problème de discontinuité, consiste à ajouter une pondération au terme de lissage d'une façon que le poids du lissage se réduit au niveau de contours en utilisant une formule décroissante en fonction du module de gradient de l'image [93].

En se basant sur l'idée que les discontinuités de l'image ne coïncident pas forcément avec les discontinuités de flot optique, le lissage a été contrôlé par les modules des dérivées de

TABLE I.4. : Quelques pénalisations robustes

Pénalisations	**Équations**
La norme L^2	$\varphi(z) = z^2$
La norme L^1	$\varphi(z) = \|z\|$
L1 modifié	$\varphi(z) = \sqrt{z^2 + \varepsilon^2}$
Logarithmique	$\varphi(z) = \log(z)$
Lorentzian	$\varphi(z) = \log(1 + \frac{z^2}{2.\sigma^2})$

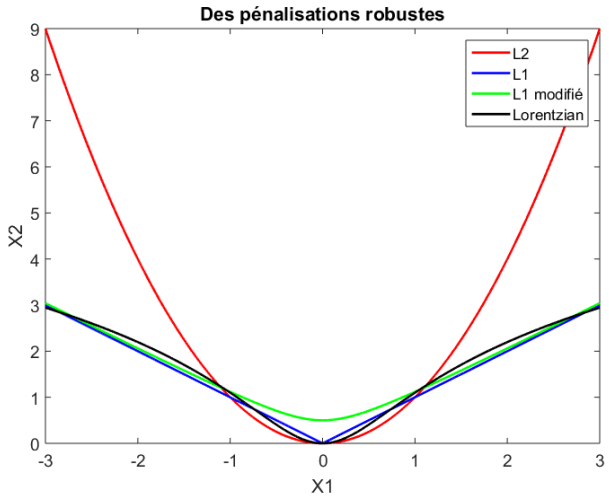


FIGURE I.7. : Les courbes de quelques pénalisations robustes

flot optique et non pas de l'image [63, 94].

Ces deux dernières stratégies de lissage ont été combinées [95] pour améliorer les résultats. Le lissage isotropique a été remplacé par un lissage anisotropique pour conserver mieux les discontinuités [96, 97].

On trouve aussi les méthodes de flot optique hybride globale-locale qui visent à obtenir un flot optique dense, moins sensible au bruit et au même temps calculer les mouvements locaux [89]. La formule combinée globale-locale peut être donnée comme suit :

$$E(\omega(\mathbf{x})) = \int (\sum_{W} W(x,y)(I_t + \nabla I.\omega) + \gamma.(\| \nabla u \|^2 + \| \nabla v \|^2))d\mathbf{x} \qquad \text{(I.18)}$$

Cette fonctionnelle est composée de deux termes, un terme de données local correspond à

l'approche de Lucas et Kanade et un terme de lissage global correspond à l'approche Horn et Schunck.

I.3.1.3 Des méthodes récentes

La littérature récente sur les méthodes de flot optique présente une variété de méthodes. Dernièrement, les méthodes d'optimisation combinatoire comme graph cuts et belief propagation ont été utilisées pour la minimisation d'un modèle discret de flot optique basé sur le champ du Markov aléatoire, ce qui permet de travailler avec n'importe quel terme de données ou de lissage [98, 99, 26].
Les méthodes de flot optique s'appuient sur le développement de Taylor de l'équation de flot optique en supposant que les déplacements des pixels sont petits. Il est possible à repérer les grands déplacements avec l'implémentation pyramidale (voir figure. I.8). Cette implémentation consiste à représenter l'image initiale par une pyramide. Les images des niveaux supérieurs sont les images de faible résolution, chaque image étant obtenue à partir du niveau inférieur, par filtrage passe-bas suivi d'un sous-échantillonnage. À chaque niveau, en commençant par le niveau de résolution la plus faible situé au sommet de la pyramide, le mouvement est estimé, puis transmis au niveau de résolution supérieure où il est utilisé comme initialisation. Cette dernière est alors raffinée par n'importe quelle méthode d'estimation de mouvement. Le processus étant repris pour le niveau suivant, et ainsi de suite jusqu'au niveau de résolution qui correspond à l'image initiale où on récupère le flot optique final [24, 78, 100].

Malheureusement, l'interpolation utilisée dans l'implémentation pyramidale non seulement enlève certaines informations qui sont importantes pour établir une bonne estimation de mouvement, mais elle ne peut également pas calculer les mouvements de petits objets. Ce problème peut être résolu par l'intégration des techniques de mise en correspondance des caractéristiques locales dans les méthodes de flot optique [101, 102, 103].
Les travaux récents sur le calcul de flot optique s'intéressent aussi à la résolution des défis majeurs qui apparaissent dans les scénarios réels et les scènes en plein air, tels que les grands déplacements, discontinuités de mouvement, changements d'illumination et les occlusions comme [67, 68, 69, 70, 5, 71].

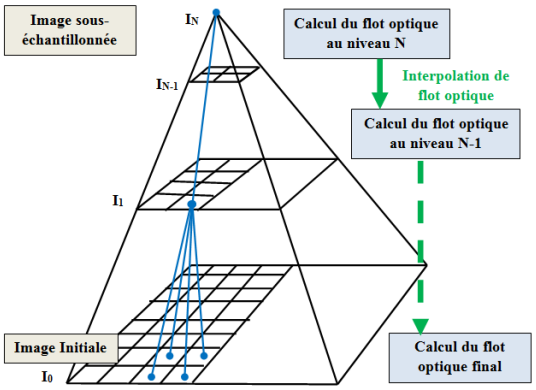


FIGURE I.8. : Implémentation pyramidale

I.3.1.4 Inconvénients des méthodes de flot optique

Les méthodes de flot optique procurent un flot optique dense en calculant un vecteur de mouvement pour chaque pixel. Cependant, ces méthodes souffrent d'un nombre de problèmes :

- La complexité de calcul est élevée. D'une part, la considération de deux contraintes pour la résolution du problème d'EM exige un temps de calcul excessive. D'autre part, les méthodes de résolution utilisées comme Newton et Gauss Seidel sont très lentes

- Le nombre des vecteurs de flot optique est égal au nombre des pixels dans l'image. Pour envoyer ces vecteurs, à un décodeur par exemple, une quantité d'information très élevée sera transmise.

- Ces méthodes se basent sur le développement de Taylor de l'équation de flot optique en supposant que les mouvements des pixels sont petits. Donc, ces méthodes ne sont pas très appropriées pour les larges déplacements,

- La contrainte de régularisation ou lissage utilisée n'est pas robuste pour les discontinuités.

I.3.2 Les méthodes de mise en correspondance de blocs (à base de bloc)

Les méthodes de mise en correspondance de blocs (BM : block matching) sont les méthodes d'EM les plus appropriées aux applications temps réel, puisqu'elles attribuent à chaque bloc de pixels un seul vecteur de mouvement.

I.3.2.1 Le concept de base de BM

L'idée générale de cette méthode consiste à diviser l'image courante en blocs non chevauchés de tailles $B_x \times B_y$ (souvent 16×16, 8×8 ou 4×4). Ensuite, pour chaque bloc l'algorithme cherche du bloc le plus similaire dans l'image de référence. Le vecteur de mouvement calculé est la différence entre la position du bloc courant et le bloc le plus similaire. Pour éviter les correspondances incohérentes dans l'image de référence, la recherche est limitée par une fenêtre de dimensions $(2 \times w_{max} + 1) \times (2 \times w_{max} + 1)$, où w_{max} est la valeur maximale du déplacement à estimer (voir figure. I.9) [24, 104].

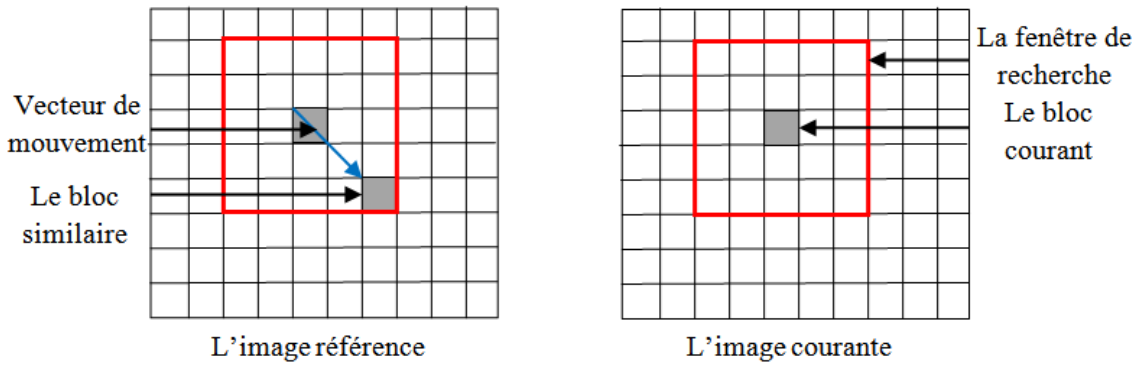


FIGURE I.9. : Le principe de base de BM

Les critères de similarité

Pour déterminer le bloc le plus similaire dans l'image de référence, plusieurs critères de similarité peuvent être utilisés [79, 22] :

- **Mesures de disparité**

  Somme des différences absolues (SAD) :

$$SAD = \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} |I_1(x+i,y+j) - I_2(x+i+u,y+j+v| \tag{I.19}$$

Somme des différences carrées (SSD) :

$$SSD = \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} (I_1(x+i,y+j) - I_2(x+i+u,y+j+v))^2 \tag{I.20}$$

- **Mesures de ressemblance**

  Coefficient d'inter-corrélation (CC) :

$$CC = \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} (I_1(x+i,y+j).I_2(x+i+u,y+j+v)) \tag{I.21}$$

  Coefficient d'inter-corrélation normalisé et centré (NCC) :

$$NCC = \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} \frac{((I_1(x+i,y+j) - \bar{I_1}).(I_2(x+i+u,y+j+v) - \bar{I_2})}{\sum_{i=1}^{B_x} \sum_{j=1}^{B_y} (I_1(x+i,y+j) - \bar{I_1})^2 . \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} (I_2(x+i+u,y+j+v) - \bar{I_2})^2} \tag{I.22}$$

  Où $\bar{I_1}$ et $\bar{I_2}$ représentent la moyenne des intensités des pixels des blocs dans l'image I_1 et I_2 respectivement.

  Peak signal to noise ratio PSNR :

$$PSNR = 10 \log_{10}(\frac{I_{max}^2}{MSE}) \tag{I.23}$$

  Où

$$MSE = \frac{SSD}{B_x \times B_y} \tag{I.24}$$

I.3.2.2 Les propriétés de BM

Les méthodes de BM sont adaptables aux applications en temps réel. En effet, d'une part leur implémentation est simple, rapide et efficace, d'autre part, la quantité d'information à transmettre est très réduite.

En plus, elles sont contrôlables. D'une part, la taille du bloc de pixels peut être fixée selon

l'application. Pour des applications qui demandent une précision d'estimation de mouvement élevée, la taille du bloc peut être petite, mais la complexité du calcul va augmenter. Tandis que, pour les applications où la complexité du calcul est plus importante que la précision, la taille du bloc peut-être grande. Si la taille du bloc est 1×1, nous obtenons les méthodes d'estimation de mouvement à base de pixel. D'autre part, la taille de la fenêtre peut être réglée selon le type de mouvement dans la séquence vidéo. Pour les séquences avec des mouvements rapides, où les objets changent leurs positions d'une image à l'autre avec une distance considérable, la taille de la fenêtre de la recherche doit être grande, engendrant bien sûr un temps de calcul plus élevé. Par contre, pour les séquences avec des mouvements faibles, on peut profiter du temps de calcul avec une taille de fenêtre plus petite [104, 24, 105]. Les méthodes BM ont été largement utilisées dans des applications diverses, le tableau. I.5 présente quelques-unes.

TABLE I.5. : Quelques applications de BM

Domaine d'application	**Référence**
Compression vidéo	Série MPEG : [106] [107] et [108]
	Série H.26X : [109], [110] et [111]
Suivi d'objet	Lai, et al. (2018) [112]
Débruitage des vidéos	Xiao, et al (2018) [113]
Stabilisation d'images	Yong-xiang, et al. (2010) [114]
Segmentation	Gamino-Sánchez, et al. (2018) [115]
Communication	Nijad, et al. (2017) [116]
La météorologie	Dazhi, et al. (2013) [117]
Détection de la peau	Kaabneh, et al. (2007) [118]
Télé-médecine	Zoha, et al. (2011) [119]

La première méthode développée pour BM adopte une stratégie de recherche exhaustive dans l'image de référence (FSA : full search algorithm) [7]. Pour trouver le bloc le plus similaire, tous les blocs candidats dans la fenêtre de recherche sont évalués (voir figure. I.10), de cette façon le bloc le plus similaire est trouvé sûrement.
Cette stratégie de recherche nécessite l'évaluation d'un nombre de blocs égal à $(2 \times w_{max} + 1)^2$. Le tableau. I.6 donne le nombre de blocs évalués par FSA en fonction du déplacement maximal. Le nombre de blocs augmente exponentiellement avec w_{max}, et également la complexité

de calcul. Par exemple avec une taille de la fenêtre de recherche $w_{max} = 7$, l'algorithme évalue $15 \times 15 = 225$ blocs. Ce nombre d'opérations demande un temps excessif qui est inacceptable pour les implémentations en temps réel, pour cela, plusieurs méthodes de BM rapides ont été mises au point, le tableau. I.7 présente quelques-unes.

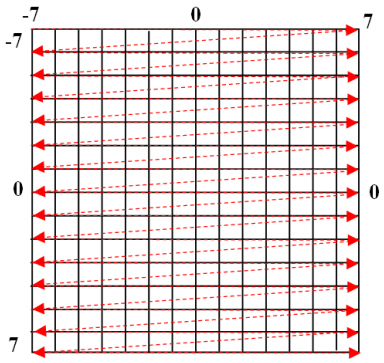


FIGURE I.10. : La recherche exhaustive de FSA

TABLE I.6. : Nombre de blocs évalués par FSA

Le déplacement maximal	**Le nombre de blocs évalués**
1	9
3	49
7	225
14	841
28	3349

I.3.2.3 Les méthodes BM rapides

Les méthodes BM rapides peuvent être regroupées en différentes classes comme suit :

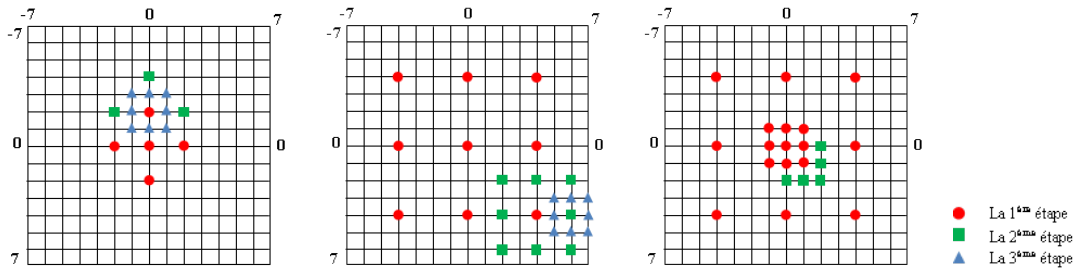
a) Les méthodes BM avec une recherche orientée

Elles consistent à une exploration initiale du voisinage direct du bloc puis une sélection dans la direction de la ressemblance croissante. Les méthodes BM orientées les plus connues

TABLE I.7. : Quelques méthodes BM

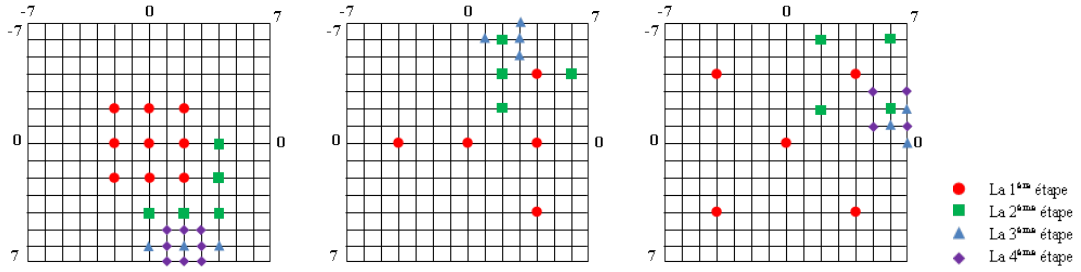
Catégorie	Méthode	Référence
Recherche exhaustive	La méthode FSA	Hsieh, et al. (1992) [7]
Les méthodes orientées	La recherche à trois étapes (TSS)	Jong et, al. (1994) [9]
	La recherche à quatre étapes (4SS)	Po et, al. (1996) [120]
	La recherche basée sur le gradient descendent BBGD	Liu et, al. (1996) [121]
	La recherche diamant (DS)	Zhu et, al. (2000) [15]
	La recherche basée sur un modèle adaptatif ARPS	Nie et, al. (2002) [122]
	La recherche avec direction orientée (EDOS)	Shinde, et al. (2018) [123]
	La recherche hexagonale (HEXS)	Raj, et al. (2018) [10]
Les recherches exhaustives rapides	L'algorithme d'élimination successive	Li, et al. (1995) [16]
	La technique de saut précoce	Huang, et al. (1996) [17]
Simplification du critère de similarité	La réduction de la précision des pixels	Feng, et al. (1995) [124]
		Lee, et al. (1998) [125]
	Le sous-échantillonnage des blocs	Liu, et al. (1993) [126]
		Cheung 1997, et al. [127]
Les méthodes multi-résolution		Nam-Kwon, et al. (1995) [128]
		Chalidabhongse, et al. (1997) [129]
		Lee, et al. (1997) [130]
Les métaheuristiques	L'algorithme génétique	Chow, et al. (1993) [131]
	L'optimisation par essaim particulaire	Ren, et al (2006) [132]
		Yuan, et al. (2008) [133]
		Cai, et al. (2012) [11]
		Pandian, et al. (2013) [18]
	L'évolution différentielle	Cuevas, et al. (2013) [20]
	La colonie d'abeille artificielle	Cuevas, et al. (2013) [19]
	La recherche harmonique	Diaz-Cortés, et al. (2017) [12]
	L'algorithme Sinus Cosinus	Dash, et al. (2018) [134]
	L'algorithme hybride HS-DE	Bhattacharjee, et al. (2018) [13]
	L'algorithme hybride ABC-DE	Bhattacharjee, et al. (2018) [13]
Autres	Le filtre Kalman	Kuo, et al. (1996) [135]
	La logique floue	Chen, et al. (2001) [136]
	L'utilisation des contours et caractéristiques	Zhao, et al. (2017) [137]
	Les blocs de tailles adaptatives	Lin, et al. (2018) [138]

sont : la recherche à trois étapes (TSS : Three step search) [9], la nouvelle recherche à trois étapes (NTSS : New three step search) [139], la recherche à quatre étapes (4SS : Four step search) [120], l'algorithme de recherche orthogonale (OSA : Orthogonal search algorithm) [14], l'algorithme de recherche croisée (CSA : Cross search algorithm) [140], la recherche diamant (DS : Diamond search) [15], la recherche par zone circulaire (CZS : Circular zonal search) [141], la recherche un à la fois (OTS : One-at-a-time search) [142], logarithmique bi-dimensionnel (2DLOG : Two dimensional logarithmic) [143], la recherche basée sur un modèle adaptatif (ARPS : Adaptive rood pattern search) [122] et la recherche basée sur le gradient descendent (BBGD : Block-based gradient descent search) [121]. Ces méthodes peuvent effectivement accélérer le processus de BM, cependant elles peuvent échouer dans le cas d'existence de plusieurs optima locaux et calculent par conséquent des vecteurs de mouvements incorrects [20, 34]. Les figures I.11, I.12 et I.13 illustrent quelques exemples de ces méthodes.



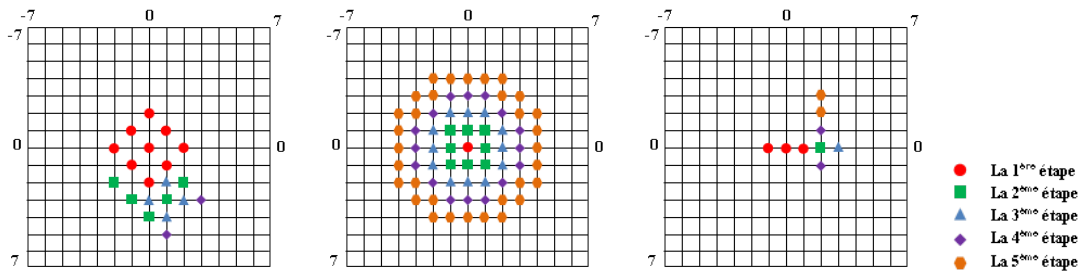
(a) 2DLOG (b) TSS (c) NTSS

FIGURE I.11. : Exemples des méthodes BM rapides : 2DLOG, TSS et NTSS



(a) 4SS (b) OSA (c) CSA

FIGURE I.12. : Exemples des méthodes BM rapides : 4SS, OSA et CSA



(a) DS (b) CZS (c) OTS

FIGURE I.13. : Exemples des méthodes BM rapides : DS, CZS et OTS

b) Les méthodes BM avec une recherche exhaustive rapide

Ces méthodes évaluent tous les blocs candidats mais en évitant les calculs non nécessaires. Parmi ces méthodes :

- L'algorithme d'élimination successive : cette méthode établit une contrainte d'inégalité, et s'intéresse seulement aux blocs candidats qui satisfassent cette contrainte [16].
  Soit le bloc dans l'image courante C et le bloc candidat dans l'image référence R.

$$S_C - SAD_{min} \leq S_R \leq S_C + SAD_{min} \tag{I.25}$$

  Où
  S_C est la somme des intensités des pixels du bloc C, S_R est la somme des intensités des pixels du bloc R et SAD_{min} est la valeur de dis-similarité obtenue jusqu'à ce bloc candidat.

- La technique de saut précoce : cette méthode a été proposée pour accélérer l'évaluation des blocs candidats. La valeur de disparité entre le bloc dans l'image courante et le bloc candidat est calculé pixel après pixel. Une fois que la somme cumulée de ces valeurs est plus élevée qu'une valeur de seuil particulière, l'évaluation de ce bloc est arrêtée, et l'algorithme sort de la boucle des pixels, car il est impossible pour ce bloc candidat d'être le bloc le plus similaire [17].

c) Les méthodes BM avec simplification du critère de similarité

Dans cette catégorie de méthodes, deux types d'algorithmes peuvent être distingués :

- La réduction de la précision des pixels : puisque les images de faible résolution tonale nécessitent une complexité de calcul réduite, la résolution originale de huit bits peut être réduite en tronquant les bits moins significatifs.

- Le sous-échantillonnage des blocs : seulement une partie sélectionnée des pixels des blocs est employée pour le calcul de la valeur de similarité. Différents types de sous-échantillonnage peuvent être appliqués, (voir figure I.14)

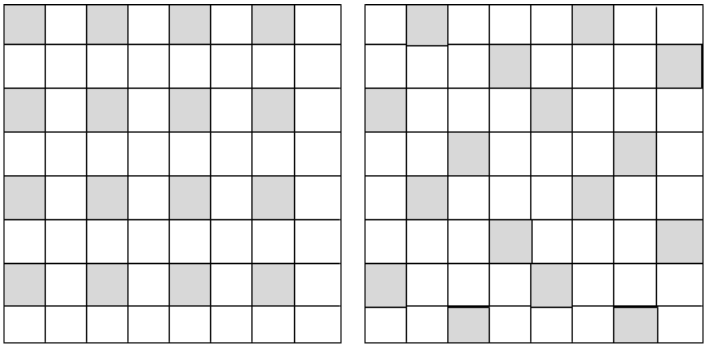


FIGURE I.14. : Exemples des méthodes BM avec simplification de critère de similarité

d) L'implémentation pyramidale

Cette technique a été proposée dans [130] pour éliminer quelques blocs candidats. Premièrement, des structures pyramidales, pour le bloc dans l'image courante et les blocs candidats sont construites comme illustré sur la figure. I.15. La somme de chaque 2×2 pixels voisins représente la valeur du pixel dans le niveau supérieur. L'algorithme commence par mesurer la valeur SAD_0 du niveau supérieur.

$$SAD_0 = \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} |C_0^{i,j} - R_0^{i,j}| \tag{I.26}$$

Avec C et R sont le bloc dans l'image courante et un bloc candidat dans l'image référence respectivement.
Si la valeur SAD_0 est plus grande qu'une valeur SAD minimale, SAD_{min}, ce bloc ne représente pas une solution potentielle et il est éliminé, sinon, le deuxième niveau est vérifié. L'algorithme continue à vérifier les niveaux du pyramide jusqu'à ce qu'on élimine ce bloc ou le niveau inférieur est évalué. Si le niveau inférieur est vérifié et sa valeur SAD_3 est plus inférieur que la valeur SAD_{min}, la valeur SAD_{min} est remplacée par SAD_3.

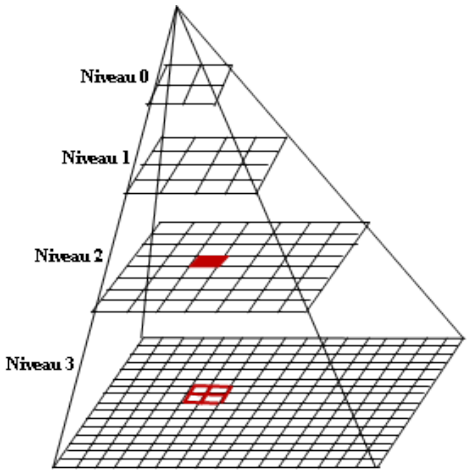


FIGURE I.15. : La méthode BM avec une implémentation pyramidale

e) Des algorithmes BM avec des tailles adaptatives de la fenêtre de recherche

La taille de la fenêtre de recherche a un effet direct sur la qualité d'estimation de mouvement et la complexité de calcul. Pour effectuer une recherche exhaustive et rapide, différentes techniques BM avec un ajustement adaptatif de la taille de la fenêtre de recherche ont été proposées. Par exemple, dans [144] les auteurs ont supposé que le champ de mouvement est lisse et change lentement entre les images, et ils ont calculé une valeur prédite pour le vecteur de mouvement de chaque bloc. Cette valeur prédite est la moyenne des vecteurs de mouvement des blocs voisins (voir figure.I.16), en se basant sur le fait qu'il existe une grande corrélation entre les vecteurs de mouvement des blocs adjacents s'ils appartiennent au même objet. Une fenêtre de recherche carrée est ensuite déterminée selon $VM_{predite}$.

$$VM_{predite} = VM_1 + VM_2 + VM_3 + VM_4 \tag{I.27}$$

f) Les méthodes BM basées sur les métaheuristiques

Récemment, le problème de BM a été posé comme un problème d'optimisation, et avec l'avènement des techniques métaheuristiques, plusieurs techniques métaheuristiques ont été utilisées pour le résoudre. Contrairement à l'algorithme FSA qui évalue tous les blocs candi-

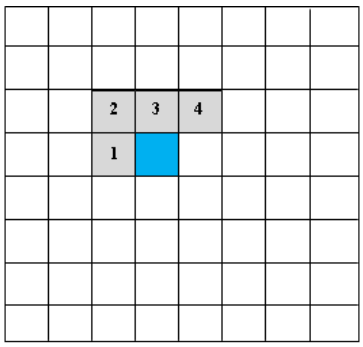


FIGURE I.16. : Les blocs voisins utilisés pour calculer le vecteur de mouvement prédit

dats dans la fenêtre de recherche, les algorithmes de BM à base de métaheuristiques initialisent seulement quelques blocs candidats dans la fenêtre de recherche, ensuite des processus aléatoires intelligentes sont utilisés pour converger itérativement vers le bloc le plus similaire (voir figure. I.17).

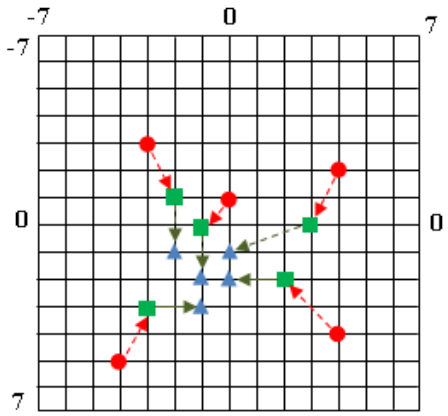


FIGURE I.17. : Le principe des méthodes BM basées sur les métaheuristiques

I.3.2.4 Évaluation des performances des méthodes BM

Les performances de chaque algorithme BM sont évaluées selon les critères suivants :

1. **La précision de l'estimation de mouvement** : La valeur de D_{PSNR} mesure le pourcentage de dégradation en *PSNR* par rapport aux résultats obtenus avec la recherche

exhaustive FSA. Elle est donnée comme suit :

$$D_{PSNR} = -\frac{PSNR_{FSA} - PSNR_{BMA}}{PSNR_{FSA}} \times 100 \tag{I.28}$$

Le $PSNR$ indique la qualité de l'image compensée qui est construite avec les blocs les plus similaires trouvés à l'aide de l'algorithme de BM.

2. **Complexité de calcul** : on peut mesurer le temps de calcul mis par l'algorithme, mais cette mesure est dépendante de la machine et le langage utilisé, pour cela, nous comparons la complexité de calcul des algorithmes BM en terme du nombre moyen des blocs évalués.

Chaque algorithme BM vise à obtenir une valeur minimale de D_{PSNR} et un nombre moyen de blocs évalués réduit.

I.4 Conclusion

Dans ce chapitre, nous avons expliqué le principe d'estimation de mouvement, ensuite nous avons présenté les deux grandes catégories des techniques d'EM. Nous avons observé que les méthodes de BM sont les méthodes d'EM les plus utilisées grâce à leur implémentation simple et efficace. Pour accélérer la stratégie de recherche exhaustive, différentes méthodes de BM rapides ont été développées, parmi eux les méthodes BM basées sur les métaheuristiques. Ces dernières méthodes possèdent la robustesse, la précision et le temps d'exécution est acceptable. Elles seront expliquées davantage dans les chapitres suivants.

Chapitre II : État de l'art sur les techniques métaheuristiques

Sommaire

II.1 Introduction

Les métaheuristiques sont des méthodes d'optimisation approximatives, conçues pour résoudre les problèmes difficiles quand les méthodes exactes échouent, ou elles nécessitent un temps de calcul inacceptable. Généralement, toutes les techniques métaheuristiques sont basées sur des mécanismes aléatoires pour explorer l'espace de recherche, afin de déterminer ou d'approcher un optimum global. Les métaheuristiques ont été utilisées dans des domaines divers tel que le traitement d'images, l'électronique, mécaniques, cryptage, etc.
Nous allons présenter dans ce chapitre, le problème d'optimisation, les techniques métaheuristiques et leurs applications.

II.2 L'optimisation

De nombreux problèmes dans la recherche opérationnelle, les mathématiques appliquées, en analyse et en statistique ou encore en théorie du contrôle et de la commande, peuvent être considérés comme des problèmes d'optimisations.
Un problème d'optimisation, au sens général, consiste à déterminer une solution x qui appartient à un espace de recherche X. Cette solution minimise ou maximise une fonction objectif f. De plus, un problème d'optimisation peut présenter des contraintes d'égalité et/ou d'inégalité sur les solutions candidates ; généralement ce sont des conditions supplémentaires qui limitent l'espace de recherche [145].

Formulation mathématique

Le problème d'optimisation peut être décrit comme suit :

Trouver $x \in X$ et $x = \begin{bmatrix} x_1 \\ x_2 \\ : \\ x_n \end{bmatrix}$ qui minimise (ou maximise) $f(x)$.

Sous les contraintes :

$$g_j(x) \quad j = 1 \cdots m \tag{II.1}$$

Où les variables $x = \begin{bmatrix} x_1 \\ x_2 \\ : \\ x_n \end{bmatrix}$ sont appelées "variables d'optimisation" ou variables de décision, la fonction f est la fonction objectif (appelée aussi fonction-coût ou fonction de fitness), elle est définie selon le problème posé, et m est le nombre de contraintes.

La résolution d'un problème d'optimisation revient alors à trouver les points de minimum (ou maximum) local (ou global) de la fonction f. La figure. II.1 donne un exemple des minima et maxima locaux et globaux.

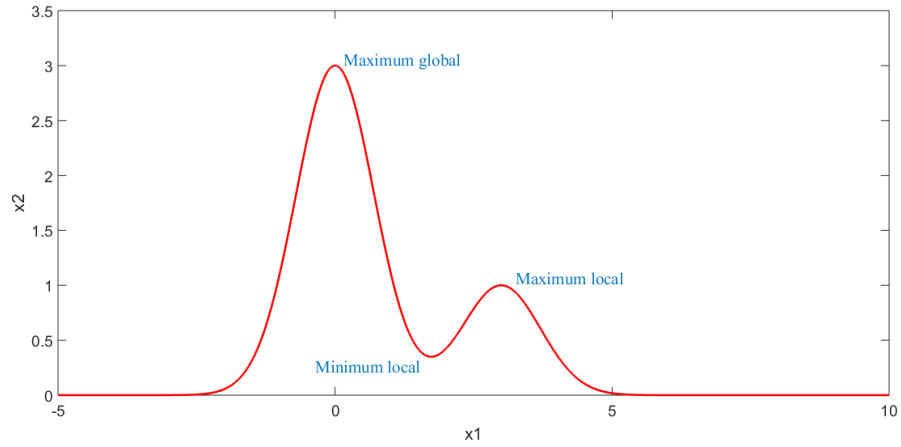


FIGURE II.1. : Les minima et maxima locaux et globaux d'une fonction

II.2.1 Classification des problèmes d'optimisation

Il existe plusieurs critères pour classifier les problèmes d'optimisation [145, 146, 147] :

- **Classification selon l'existence des contraintes** : le problème d'optimisation peut-être sans ou avec des contraintes.

- **Classification selon la nature des variables** : selon la nature des variables les problèmes d'optimisation peuvent être classifiés en deux catégories :

  1. Dans la première, le problème consiste à trouver la valeur optimale d'un paramètre, par exemple :

     Trouver $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ qui minimise (ou maximise) $f(x)$.

  2. Dans la deuxième, le problème consiste à trouver un ensemble des valeurs d'un paramètre qui est en fonction d'un autre paramètre, par exemple :

Trouver $x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}$ qui minimise (ou maximise) $f\{x(t)\}$

Ce type de problème où chaque variable est une fonction d'un ou plusieurs paramètres, est connu comme problème d'optimisation dynamique ou trajectoire.

- **Classification selon la nature des équations impliquées** : cette classification regroupe les problèmes d'optimisation en quatre catégories : linéaire, non-linéaire, géométrique et quadratique.

  1. Linéaire : si la fonction objectif et toutes les contraintes sont des fonctions linéaires, le problème s'appelle un problème de programmation linéaire (LP).

  2. Non-linéaire : si une fonction parmi les fonctions d'objectif ou les contraintes est non-linéaire, le problème s'appelle un problème de programmation non-linéaire (NLP).

  3. Géométrique : dans le problème de programmation géométrique (GMP), la fonction objectif et les contraintes sont exprimées comme posynomial en X, (l'expression de la fonction posynomial est donnée dans l'annexe A).

  4. Quadratique : si la fonction objectif est quadratique et les contraintes sont linéaires, on dit que le problème est quadratique, habituellement, il est formulé comme suit :

$$f(x) = c + \sum_{i=1}^{n} q_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{i,j} x_i x_j \tag{II.2}$$

  Sous les contraintes :

$$\sum_{i=1}^{n} a_{i,j} x_i = b_j \tag{II.3}$$

  Où c, q_i, Q_{ij}, b_j et a_{ij} sont des constantes.

- **Classification selon le type des variables** : les variables du problème d'optimisation prennent des valeurs continues ou discrètes, et par conséquence, on trouve des problèmes d'optimisation continus et d'autres discrets.

- **Classification selon la nature déterministe des variables** : si certaines ou toutes les variables d'un problème d'optimisation sont probabilistes (non-déterministes ou stochastique), le problème est de programmation stochastique, sinon le problème est de programmation déterministe.

- **Classification selon le nombre de fonctions objectifs** : selon le nombre de fonctions objectifs à minimiser (ou maximiser), le problème d'optimisation peut être mono- ou multi-objectif.

II.2.2 Les techniques d'optimisation

Les techniques d'optimisation sont nombreuses et on peut les classifier en deux catégories selon qu'elles incluent ou non l'aspect aléatoire : les techniques déterministes (exactes) et les techniques non-déterministes (approchées) (voir figure.II.2) [148, 149, 150, 162].

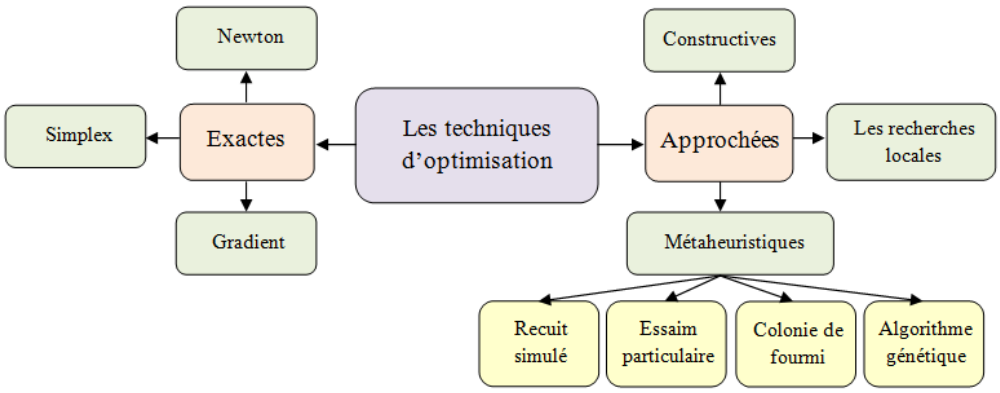


FIGURE II.2. : Classification des techniques d'optimisation

II.2.2.1 Les techniques exactes

Elles ont été introduites pour la résolution des problèmes d'optimisation d'une manière exacte, elles garantissent l'obtention de la solution optimale absolue, pour certains types de problèmes d'optimisation. Parmi ces méthodes, on peut citer la méthode de newton, la méthode du simplex ou encore la méthode du gradient, etc. Souvent, l'utilisation de ces méthodes est couronnée de succès. Cependant, plusieurs inconvénients limitent leur emploi :

- Les méthodes exactes nécessitent que la fonction objectif soit convexe, continue et dérivable ;

- Elles sont utilisables tant que le nombre de variables est petit ;

- La fonction d'objectif peut avoir de multiples minima locaux. Or les méthodes exactes sont incapables de détecter et d'éviter ceux-ci, en convergeant vers le premier optimum trouvé qui correspond à une valeur de fitness médiocre, en comparaison de celui de optimum absolu.

- Le temps de calcul est souvent trop élevé.

II.2.2.2 Les techniques approchées

Dans le cas où les méthodes exactes échouent à résoudre un problème d'optimisation ou elles nécessitent un temps de calcul très élevé, il est probablement nécessaire d'avoir recours aux techniques approchées.
Les techniques approchées ou heuristiques procurent une solution approchée, sans garantie l'optimalité, au profit d'un temps de calcul réduit, parmi ces méthodes, on trouve les méthodes constructives, les recherches locales et les métaheuristiques.

Les métaheuristiques sont des heuristiques plus poussées, le terme métaheuristique a été inventé par Fred Glover en 1986, il est composé en deux mots : '*heuristique*' qui signifie 'trouver' et le suffixe '*méta*' qui signifie 'au-delà' ou encore 'dans un niveau supérieur'. La principale qualité de ces méthodes est qu'elles sont adaptables à un grand nombre de problèmes d'optimisation. Généralement, ces méthodes commencent par un tirage aléatoire d'un nombre de solutions candidates, ces dernières changent leurs positions itérativement avant qu'elles convergent à une solution optimale (voir figure.II.3). Nous continuerons, dans la suite de cette thèse à nous intéresser uniquement aux techniques métaheuristiques et leurs applications.

II.3 Les métaheuristiques

Les métaheuristiques sont des algorithmes d'optimisation stochastiques conçus pour résoudre les problèmes d'optimisation difficiles. Elles ont été utilisées avec succès dans des applications diverses grâce à leur simplicité et robustesse. D'après les études menées par Ka-

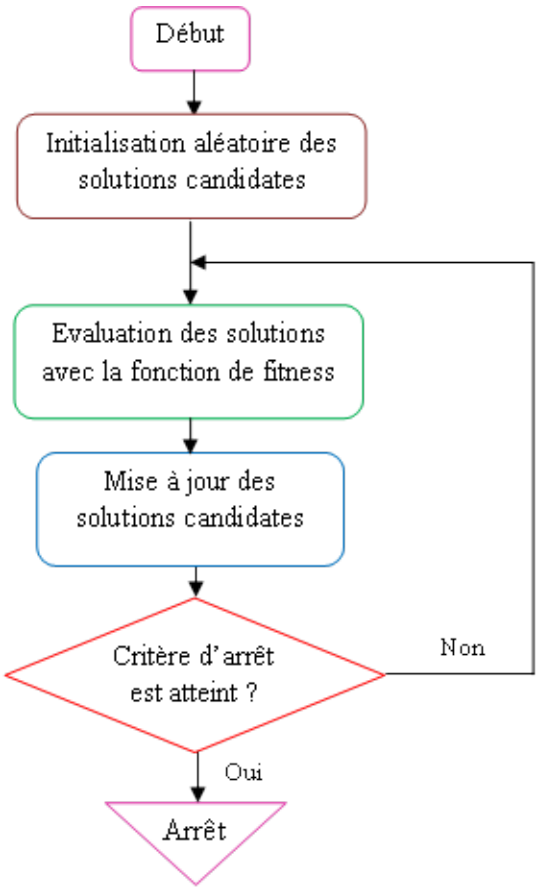


FIGURE II.3. : Le schéma général des techniques métaheuristiques

shif, et al. en 2018 [21], il y a presque 1222 publications sur les métaheuristiques, répartis en différents types. Le tableau. II.1 et le graphique présenté dans la figure. II.4 donnent le nombre des travaux publiés sur les métaheuristiques par des journaux différents.

Généralement, toutes les techniques métaheuristiques ont des caractéristiques communes qui sont : [148, 150, 149]

- Elles commencent par un tirage aléatoire d'un nombre de solutions candidates, ensuite, des processus de recherche aléatoire sont utilisés pour manipuler ces solutions et les faire passer de solutions de mauvaise qualité à la solution optimale.

- Les métaheuristiques sont des algorithmes itératives, où le même schéma de recherche est répété plusieurs fois au cours de l'optimisation.

TABLE II.1. : Les types de publications sur les métaheuristiques

Journal	Type de publication						Nombre total
	Nouvelles méthodes	Méthodes modifiées	Méthodes hybrides	Études	Comparaison et analyse	Applications	
Elsevier	54	77	26	55	54	43	309
Springer	28	61	24	85	37	38	273
Hindawi	14	96	29	11	14	49	213
IEEE Xplore	16	51	18	7	21	65	178
ACM	5	47	9	27	23	8	119
Other	10	11	0	24	8	16	69
Wiley	3	7	3	31	5	12	61

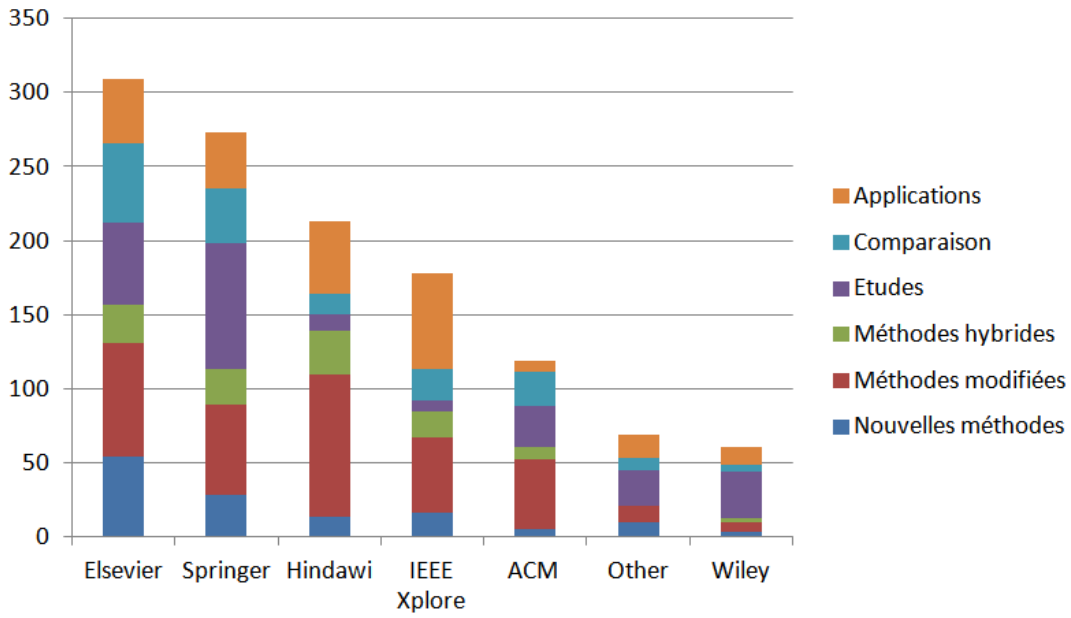


FIGURE II.4. : Graphe sur les publications en métaheuristiques

- Elles n'utilisent pas l'information du gradient de la fonction objectif.
- Elles sont capables de résoudre des problèmes d'optimisation difficiles à partir d'un nombre minimal d'information.
- Elles peuvent trouver une solution de très bonne qualité en un temps raisonnable sans

garantie l'optimalité.

- La nature stochastique des métaheuristiques permet d'éviter les optima locaux.
- Elles sont adaptables à un grand nombre de problèmes d'optimisation.
- Souvent, elles sont inspirées par des phénomènes de la nature (voir figure. II.5).

(a) Algorithme génétique (b) Optimisation par colonie d'abeilles (c) Optimisation par colonie de fourmis

(d) Trou noir (e) Algorithme de chauve-souris (f) Optimisation par essaim particulaire

FIGURE II.5. : Quelques sources d'inspiration des techniques métaheuristiques

La résolution satisfaisante d'un problème d'optimisation en utilisant une technique métaheuristique dépend de sa capacité à fournir un bon équilibre entre l'exploration (diversification) et l'exploitation (intensification) [151].

- L'exploration est la capacité de l'algorithme à visiter différentes régions dans l'espace de recherche, pour éviter les optima locaux et la convergence prématurée.
- L'exploitation est la capacité de l'algorithme à concentrer la recherche dans les régions prometteuses de l'espace de recherche (voir figure. II.6).

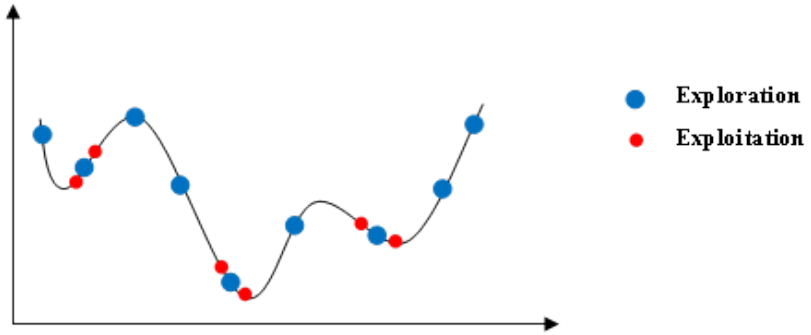


FIGURE II.6. : L'exploration et l'exploitation des métaheuristiques

II.3.1 Classification des métaheuristiques

La liste des techniques métaheuristiques est assez large, nous pouvons les classer de plusieurs manières selon le nombre de solutions, selon l'historique de recherche ou selon l'inspiration (voir figure. II.7).

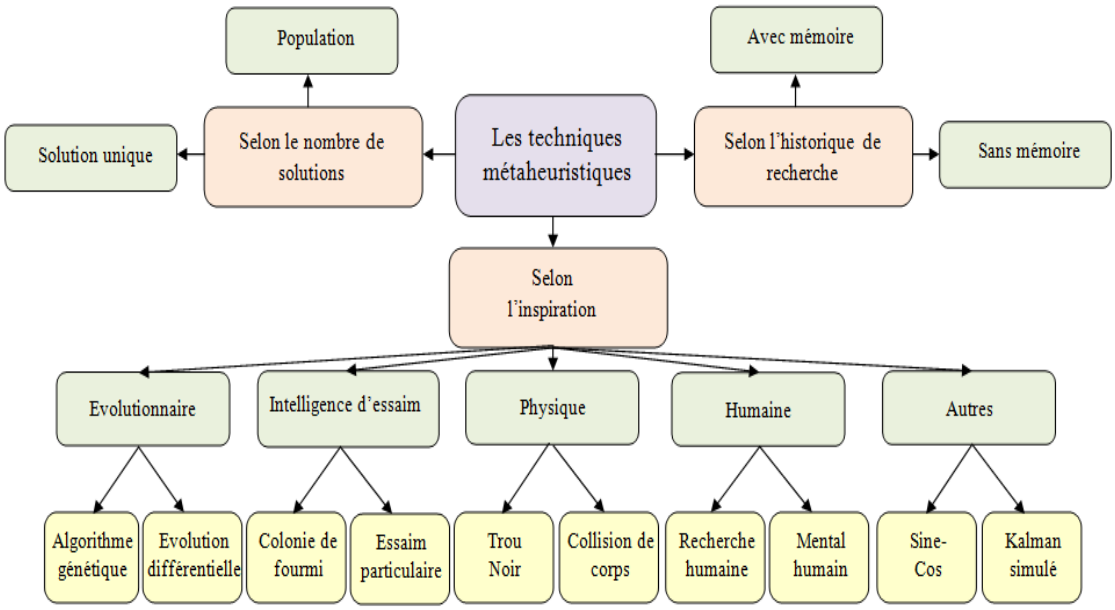


FIGURE II.7. : Classification des métaheuristiques

Selon le nombre des solutions candidates, on trouve :

- Les méthodes à base d'une solution unique, aussi appelées méthodes de trajectoire.

Elles commencent par une seule solution initiale et s'en éloignent progressivement en construisant une trajectoire dans l'espace de recherche. Parmi ces méthodes : la méthode du recuit simulé [152], la méthode de descente [153], la recherche tabou [154], la méthode GRASP [155], la recherche à voisinage variable [156] et la recherche locale itérée [157].

- Les méthodes à base d'un ensemble de solutions dite population, elles manipulent à la fois un ensemble de solutions. Parmi ces méthodes, on trouve la recherche fractale stochastique, l'optimisation par essaim particulaire, l'algorithme sine-cos, etc [158, 159, 160, 161].

Généralement, les métaheuristiques à base d'une solution unique sont plus orientées vers l'exploitation, alors que les métaheuristiques à base d'une population sont orientées vers l'exploration [162].

Selon l'utilisation de l'historique de la recherche ou non. Beaucoup de techniques métaheuristiques utilisent l'historique de leur recherche pour guider l'optimisation aux itérations suivantes, que ce soit sur le court terme (solutions visitées récemment) ou sur le long terme (mémorisation d'un ensemble de paramètres synthétiques décrivant la recherche) [163]. La méthode tabou, par exemple, garde en mémoire les dernières solutions visitées et interdit le retour vers celles-ci. La méthode d'optimisation par essaim particulaire sauvegarde la meilleure solution globale et la meilleure solution atteinte par chaque particule. Cependant, l'espace nécessaire pour enregistrer l'ensemble de solutions peut s'avérer important en espace mémoire [164].

Selon l'inspiration, il y a :

- Les algorithmes évolutionnaires (EA : Evolutionary algorithms) sont inspirés par l'évolution biologique des espèces, tels que l'algorithme génétique [165], les stratégies d'évolution [166], la programmation évolutionnaire [167], la programmation génétique [168], l'évolution différentielle [169] et la recherche fractale stochastique [158]. La plupart des EA disposent de trois processus principaux : sélection, croisement et la mutation, qui permettent le balancement entre l'exploration et l'exploitation [170]

- Les algorithmes d'intelligence en essaim (SI : Swarm intelligence) sont inspirés par

l'intelligence sociale et le comportement collectif des animaux vivant en groupes. Ils se basent sur la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie [171]. Cette catégorie englobe l'optimisation par essaim particulaire (particle swarm optimization) [172], l'algorithme de recherche de nourriture bactérienne (bacterial foraging) [173], l'algorithme de libellule (dragonfly algorithm) [159], l'optimisation des partenaires de dauphin (dolphin partner optimization) [174], les colonies de fourmis (ant colony) [175] et l'optimisation du loup gris (grey wolf optimizer) [176]. Pour assurer un bon compromis entre l'exploration et l'exploitation, ces méthodes utilisent différentes stratégies, selon la particularité de chaque méthode. Dans l'optimisation par essaim particulaire, par exemple, le cœfficient d'inertie dynamique, qui varie au cours du temps, permet de réaliser un équilibre entre la recherche locale (exploitation) et la recherche globale (exploration) [148]. L'optimisation par colonie de fourmis utilisent le phéromone et les informations heuristiques pour contrebalancer l'algorithme entre la recherche globale et locale [171].

- Les algorithmes inspirés par des phénomènes physiques tels que le trou noir [177], l'algorithme de la recherche gravitationnelle [178], l'algorithme Big Bang-Big Crunch [179], l'algorithme de recherche basé sur la galaxie [180], l'optimisation inspirée d'optique [181] et l'optimisation par collision de corps [182].

- Les techniques à base humaine sont inspirées par le comportement humain et elles modélisent les différents caractères des personnes, comme l'algorithme de la ligue des champions [183], l'optimisation basée sur l'apprentissage de l'enseignement [184], l'algorithme d'explosion de mine [185] et la recherche à base du mental humain [160].

- Autres inspirations : on trouve aussi quelques techniques inspirées par d'autres sources, comme le filtre de kalman simulé [186] et l'algorithme sinus Cosinus [161].

Le théorème de "no free lunch" (NFL) [187] a démontré que si l'on considère l'ensemble de tous les problèmes d'optimisation possibles, alors aucun algorithme n'est meilleur qu'un autre. En effet, la comparaison d'algorithmes métaheuristiques ne peut avoir lieu qu'une fois on a précisé le problème traité [149]. Selon [188] l'optimisation est non seulement une théorie mathématique mais il y a l'expérience qui guide l'utilisateur dans le choix de l'algorithme à implanter.

D'après les résultats récents d'application des métaheuristiques en différents domaines, il a été démontré que l'utilisation des algorithmes évolutionnaires procurera de nombreux d'avantages, et ils obtiennent de meilleurs résultats que les autres métaheuristiques, dans le cas des problèmes très complexes. Cette supériorité vient du fait que des solutions distinctes de la population sont sélectionnées au hasard et ensuite combinées pour créer de nouvelles solutions. La combinaison pondérée de bonnes solutions partielles peut produire de très bons résultats globaux. Le problème de ces méthodes, c'est que l'utilisation de plusieurs processus de mise à jour peut augmenter le nombre d'évaluations de la fonction de fitness.

II.3.2 Quelques techniques métaheuristiques

Dans cette partie, nous donnons le principe général de quelques méthodes métaheuristiques.

II.3.2.1 Recuit simulé

La méthode du recuit simulé, proposée par Kirkpatrick et, al. [152], est inspirée par un processus de métallurgie (appelé le recuit) où pour atteindre les états d'énergie bas d'un solide, on chauffe celui-ci jusqu'à des températures élevées avant de le laisser refroidir lentement. Cette méthode s'appuie sur l'algorithme de Metropolis [189], qui permet de sortir des minima locaux avec une probabilité élevée si la température T est élevée, et quand l'algorithme atteint de basses températures, il permet de conserver les états les plus probables. Cette méthode peut être résumée par l'algorithme.1 [148].

Initialisation d'une configuration aléatoire et une température T,
Tant que (Le critère d'arrêt n'est pas atteint) **faire**
    Tant que (L'équilibre n'est pas atteint) **faire**
        Tirer une nouvelle configuration,
        Appliquer la règle de Metropolis,
        Décroître la température T.
    Fait
Fait

Algorithme 1 : L'algorithme de la méthode de récuit simulé

II.3.2.2 La recherche tabou

La méthode de recherche tabou a été proposée par Fred Glover [154]. L'idée principale de cette technique consiste à explorer le voisinage d'une position donnée en effectuant des déplacements qui n'améliorent pas forcément la solution. Pour éviter le retour en arrière, elle utilise le principe de mémoire pour sauvegarder les solutions interdites, appelée "liste tabou ". Cette liste est mise à jour itérativement pour passer l'algorithme de l'exploration à l'exploitation. Cette méthode peut être résumée dans l'algorithme 2.

Initialisation d'une configuration aléatoire et une liste vide de tabou,
Tant que (Le critère d'arrêt n'est pas atteint) **faire**
    Perturbation de la configuration,
    Évaluation de voisinage,
    Mise à jour de la meilleure solution,
    Mise à jour de la liste tabou.
Fait

Algorithme 2 : L'algorithme de la recherche tabou

II.3.2.3 L'algorithme génétique

L'algorithme génétique est une technique évolutionnaire proposée par John Holland et ses élèves [165]. Cet algorithme dispose de trois opérations principales :

- L'opérateur de sélection pour déterminer les meilleures solutions appelées aussi parents, qui sont utilisées pour engendrer la nouvelle génération. Il y a différentes stratégies de sélection comme la sélection par tirage à la roulette (roulette-wheel selection), la sélection par rang (ranking selection), la sélection par tournoi (tournament selection)...

- L'opérateur de croisement qui combine les caractéristiques des parents (préalablement sélectionnés) pour générer les enfants. Un exemple de croisement est illustré sur la figure. II.8.

- L'opérateur de mutation qui modifie aléatoirement une partie de l'individu, ce qui permet de maintenir une certaine diversité dans la population. Un exemple de la mutation

est illustré sur la figure.II.9

Après ces trois étapes, certains individus parents sont remplacés par d'autres individus enfants. La stratégie de remplacement la plus simple est de prendre les meilleurs individus de la population en fonction de leurs performances respectives [171].

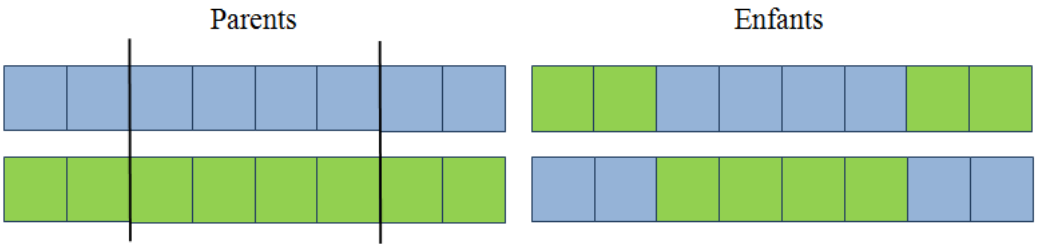


FIGURE II.8. : Exemple sur le croisement

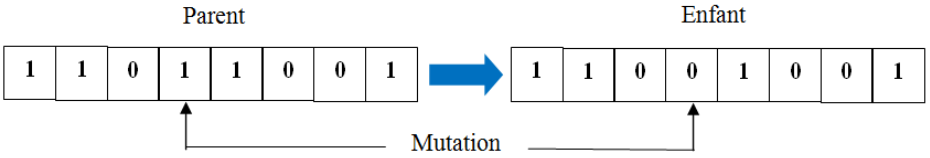


FIGURE II.9. : Exemple sur la mutation

II.3.2.4 L'optimisation par essaim particulaire

L'optimisation par essaim particulaire proposée par Russel Eberhart et James Kennedy en 1995 [172], elle est fondée sur la notion de coopération entre les animaux évoluant en essaim, tels que les bancs de poissons et les vols groupés d'oiseaux et qu'ils possèdent des capacités de perception, mémorisation et calcul limités, mais l'échange d'information entre eux fait que globalement ils arrivent néanmoins à résoudre des problèmes difficiles. Chaque particule de la population est dotée d'une vitesse, position et voisinage. Le principe de cette technique est de déplacer ces particules dans l'espace de recherche afin qu'elles convergent à l'optimum en combinant trois composantes :

- Une composante physique ; où la particule tend à suivre sa direction courante,

- Une composante cognitive ; où la particule tend à retourner à sa meilleure position,

- Une composante sociale ; où la particule tend à suivre la meilleure solution de ses voisines [148].

La vitesse et la position de chaque particule sont mises à jour à l'aide des deux équations (II.4) et (II.5) respectivement.

$$\begin{aligned} v_i(iter+1) &= \omega \times v_i(iter) \\ &+c_1 \times rand_1 \times (P_{best\,i}(iter) - x_i(iter)) \\ &+c_2 \times rand_2 \times (G_{best}(iter) - x_i(iter)) \end{aligned} \tag{II.4}$$

$$x_i(iter+1) = x_i(iter) + v_i(iter+1) \tag{II.5}$$

Où $P_{best\,i}$ est la meilleure solution personnelle de la solution i et G_{best} représente la meilleure solution globale de tout l'essaim. Les cœfficients ω, c_1 et c_2 contrôlent l'influence de chaque composante. Les deux nombres aléatoires $rand_1$ et $rand_2$ sont tirés uniformément dans l'intervalle [0, 1].

II.3.2.5 Trou noir

La méthode d'optimisation de trou noir proposée par Abdolreza Hatamlou en 2013 [177], elle est inspirée par le phénomène de trou noir. Un ensemble de solutions candidates représentent les étoiles est initialisé aléatoirement dans l'espace de recherche. À chaque itération, la meilleure solution candidate est sélectionnée pour être le trou noir, qui commence alors à tirer les autres solutions vers lui par l'équation.II.6. Si une solution s'approche trop du trou noir, elle disparaît puis elle est remplacée par une autre solution générée aléatoirement.

$$x_i(iter+1) = x_i(iter) + rand \times (x_{TN} - x_i(iter)) \tag{II.6}$$

Où x_{TN} est le trou noir, représenté par la meilleure solution globale.

II.3.2.6 L'algorithme de la recherche harmonique

L'algorithme de la recherche harmonique a été développé par Geem et, al. en 2001 [190]. Il est basé sur le processus de performance musicale qui consiste à trouver l'harmonie parfaite dans un orchestre où chaque musicien joue une note pour trouver la meilleure harmonie. D'une

manière analogue, chaque variable de décision dans le processus d'optimisation a une valeur pour trouver la meilleure solution. Quand un musicien improvise un lancement, habituellement, il suit n'importe laquelle des trois règles [191] :

- Règle 1 : jouant un lancement de sa mémoire,
- Règle 2 : jouant un lancement adjacent d'un lancement de sa mémoire,
- Règle 3 : jouant le lancement totalement aléatoire de la gamme saine et possible.

II.3.2.7 Filtre Kalman simulé

La méthode de filtre Kalman simulé proposée par Zuwairie et, al. [186], elle est inspirée par la capacité d'estimation du filtre de Kalman, où le problème de l'estimation de l'état est considéré comme un problème d'optimisation, et chaque solution candidate agit comme un filtre de Kalman. Cette méthode dispose de trois étapes principales :

- Prédiction : les deux équations.II.7 et II.8 sont utilisées pour prédire l'état et les erreurs de covariance obtenues par les estimations a priori.

$$x(iter \setminus\ iter-1) = x(iter-1) \tag{II.7}$$

$$P(iter \setminus\ iter-1) = P(iter-1) + Q \tag{II.8}$$

  Où Q est l'erreur de processus.

- Mesure : c'est une réaction au processus d'estimation. La mesure de chaque solution est simulée à l'aide de l'équation

$$Z_i(iter) = x_i(iter \setminus\ iter-1) + sin(2\pi\, rand) \times [x_i(iter \setminus\ iter-1) - x_{true}] \tag{II.9}$$

  Où x_{true} est la meilleure solution globale, et *rand* est un nombre aléatoire tiré uniformément dans l'intervalle [0, 1].

- Estimation : durant cette étape, le gain du filtre Kalman est calculé par l'équation.II.10, et ensuite utilisé pour améliorer les estimations a postériori après la mesure à l'aide des

équations.II.11 et II.12.

$$K(iter) = \frac{P(iter \setminus\ iter-1)}{P(iter \setminus\ iter-1)+R} \tag{II.10}$$

$$x_i(iter) = x_i(iter \setminus\ iter-1) + K(iter) \times [Z_i(iter) - x_i(iter \setminus\ iter-1)] \tag{II.11}$$

$$P(iter) = (1-K(iter)) \times P(iter \setminus\ iter-1) \tag{II.12}$$

Où R est le bruit mesuré et $iter$ est itération courante.

II.3.3 Extensions des métaheuristiques

Une caractéristique très intéressante des techniques métaheuristiques est qu'elles se prêtent naturellement à des extensions dont le but est d'adapter les métaheuristiques à un grand nombre de problèmes d'optimisation [148, 150, 192, 193]. Parmi ces extensions, on peut citer :

- **Multi-objectif** : ce type de méthodes ne cherche pas une solution optimale unique mais un ensemble de solutions optimales, dites Pareto optimales. Une solution est Pareto optimale si l'amélioration à l'égard d'une des fonctions objectifs entraîne invariablement une détérioration relativement à une autre fonction objectif.

- **Multi-population** : les algorithmes à plusieurs populations (ou essaims) utilisent plusieurs sous-populations pour effectuer des tâches différentes, avec la présence ou non de chevauchement (ou fusion) entre les sous-populations.

- **Parallèle** : cette implémentation est utilisée pour accélérer l'exécution par la distribution des tâches de calcul aux plusieurs processeurs.

- **Hybride** : elle consiste à combiner les techniques métaheuristiques, avec d'autres méthodes d'optimisation que ce soit exactes ou métaheuristiques afin de tirer profit des avantages respectifs.

II.3.4 Conception de nouvelles méthodes métaheuristiques

Le théorème de "No Free Lunch" (NFL) a ouvert la porte devant les chercheurs pour développer de nouvelles techniques métaheurstiques [194, 195, 196, 197, 198] ou améliorer celles qui existent [199, 200, 201, 202, 203, 245].
La conception d'une nouvelle technique métaheuristique efficace et performante n'est pas une tache facile, différentes caractéristiques essentielles doivent être prises en compte. La nouvelle technique métaheurstique doit être capable de :

- Fournir un bon équilibre entre l'exploration (pour éviter les optima locaux) et l'exploitation (pour converger rapidement vers le minimum d'une vallée donnée à partir d'un point de départ).

- Trouver un optimum robuste, c'est-à-dire obtenir une solution peu sensible aux incertitudes.

- Traiter une large gamme de problèmes d'optimisation, y compris les problèmes mono- ou multi-objectif

- L'implémentation de la méthode doit être simple et efficace, avec une complexité de calcul réduite et peu de paramètres à ajuster.

II.3.5 Applications des métaheuristiques

II.3.5.1 Les fonctions de test

Chaque nouvelle technique métaheuristique doit être validée par des fonctions de test où l'optimum global est connu a priori et le nombre d'itérations est fixé pour comparer les résultats avec d'autres techniques. Dans cette thèse, pour valider la technique métaheuristique proposée, nous nous appuyons sur vingt-trois fonctions de test définie dans le cadre de la conférence *2005 IEEE Congress on Evolutionary Computation (CEC'05)*. Le tableau dans l'annexe B fournit une description détaillée de chaque fonction de test y compris la formulation mathématique, le nombre de dimensions, l'intervalle de l'espace de recherche et le nombre maximal d'itérations. Pour les métaheuristiques à base de population, la taille de la population est toujours fixée à 100 solutions candidate pour toutes les 23 fonctions. Ces

fonctions de test sont classées en trois groupes :

- *Les fonctions uni-modales à haute dimension* $(F_{01} - F_{07})$: elles n'ont qu'une seule solution optimale globale, et elles sont conçues pour tester la capacité d'exploitation des métaheuristiques (voir figure.II.10).

- *Les fonctions multimodales à haute dimension* $(F_{08} - F_{13})$: elles sont les fonctions d'optimisation les plus difficiles, parce qu'elles ont plusieurs optima locaux et pour les surmonter et éviter la convergence prématurée une capacité d'exploration très élevée est nécessaire (voir figure.II.11).

- *Les fonctions multimodales à faible dimension* $(F_{14} - F_{23})$: elles sont similaires à la catégorie précédente, mais avec des dimensions basse, donc un nombre inférieur d'optima locaux (voir figure.II.12).

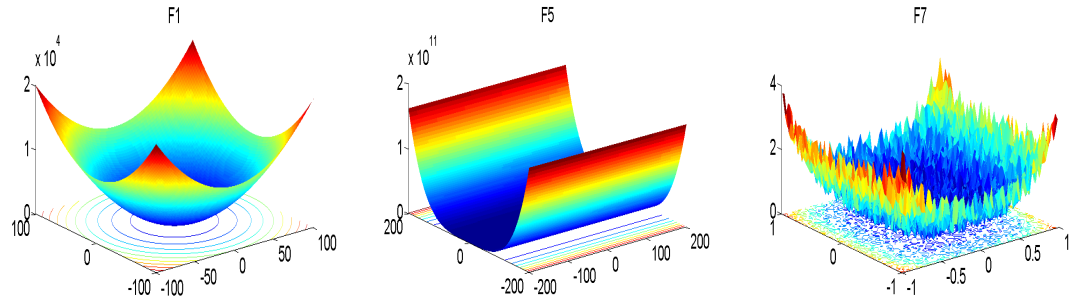


FIGURE II.10. : Exemples sur les fonctions uni-modales à haute dimension F_1, F_3 et F_7

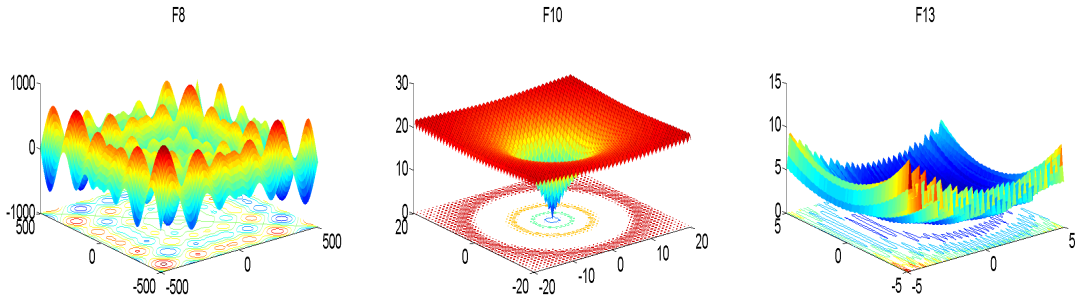


FIGURE II.11. : Exemples sur les fonctions multimodales à haute dimension F_8, F_{10} et F_{13}

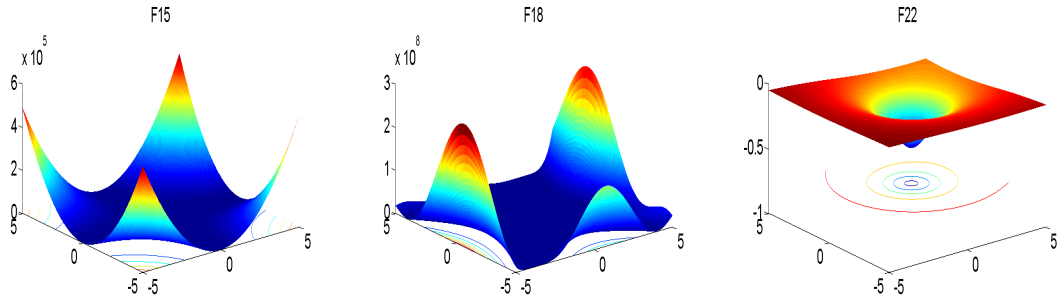


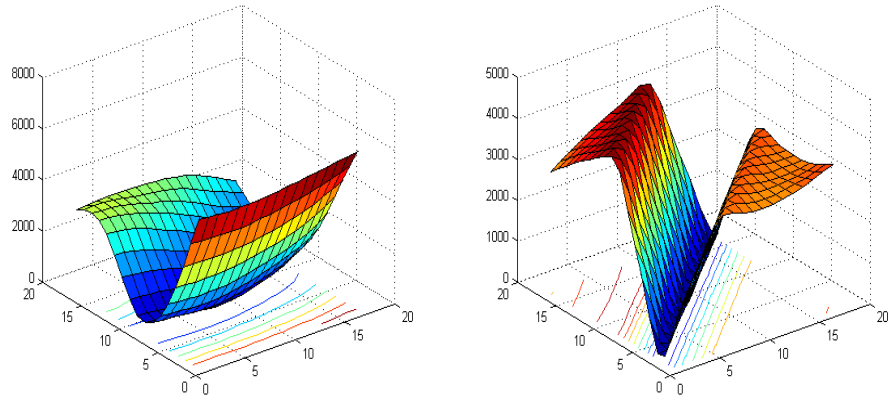
FIGURE II.12. : Exemples sur les fonctions multimodales à faible dimension F_{15}, F_{18} et F_{22}

II.3.5.2 Applications réelles

Au niveau d'application, les techniques métaheuristiques ont été appliquées avec succès dans plusieurs domaines de recherche comme : les réseaux de télécommunications, formation de cellules de production, applications médicales, systèmes électriques, voyageur de commerce, ordonnancement de projet, la sécurité des systèmes de communication, problème du sac à dos, applications en multimédia, systèmes automatisés de stockage et de déstockage, problèmes d'affectation quadratique, problèmes d'allocation de ressources, tournée de véhicule, systèmes de production reconfigurables, allocation dans les chaînes logistiques, planification des séquences d'assemblage, ordonnancement dans un Open Shop et routage dans les réseaux optiques [162].

Les techniques métaheuristiques ont été aussi appliquées en estimation de mouvement, plusieurs algorithmes de mise en correspondance des blocs à base de métaheuristiques ont été proposées [132, 133, 11, 18, 20, 19, 12, 134], dont le but est d'utiliser des stratégies de recherche intelligentes pour réduire la complexité de calcul de la recherche exhaustive (FSA).

En estimation de mouvement, nous trouvons aussi les fonctions de test uni-modales et multimodales. La figure II.13 illustre des exemples sur ces fonctions. Donc, la technique métaheuristique utilisée pour résoudre ce problème doit être capable de fournir une bonne exploration et exploitation de l'espace de recherche, pour éviter les optima locaux et converger rapidement vers la meilleure solution globale.



(a) Uni-modale (b) Multimodale

FIGURE II.13. : Exemples sur les fonctions en BM

II.4 Conclusion

Nous avons rappelé dans ce chapitre quelques notions sur l'optimisation. Puis, nous avons mis en clair les deux types des techniques d'optimisation : exactes et métaheuristiques. Le principe de métaheuristiques a été développé plus profondément. Nous avons notamment présenté les fonctions de test utilisées pour la validation des métaheuristiques et également quelques applications réelles. Nous avons également observé que les métaheuristiques ont été utilisé dans le domaine d'estimation de mouvement. Dans les deux prochains chapitres, nous donnerons notre contribution conceptuelle et expérimentale pour la résolution du problème de BM avec les techniques métaheuristiques.

Chapitre III : Approche de mise en correspondance de blocs par la technique SFS

Sommaire

III.1 Introduction

Dans le cadre de l'estimation de mouvement, de nombreuses techniques de mise en correspondance de blocs (BM : block matching) rapides ont été proposées pour réduire la complexité de calcul de la technique exhaustive FSA. Parmi eux, on trouve les méthodes de BM basées sur les métaheuristiques. D'après la littérature, les résultats de ces dernières sont meilleurs que plusieurs d'autres méthodes BM, que ce soit en terme de précision ou en terme de complexité de calcul.

Toutefois, les techniques proposées dans ce contexte ont quelques limitations, par exemple les méthodes BM basées sur la technique d'optimisation par essaim particulaire (PSO) comme [132, 133, 11, 18] souffrent du problème de la convergence prématurée de PSO, et elles peuvent tomber sur des optima locaux [204]. Il y a aussi la méthode BM basée sur l'évolution différentielle (DE) [20]. Le problème de cette méthode est qu'elle fournit des résultats de convergence non-stable dans plusieurs cas [205]. On trouve également la méthode BM basée sur la colonie d'abeilles artificielle (ABC) [19]. Malgré que l'ABC est une méthode simple et robuste, mais elle effectue une convergence trop lente [206]. Il existe aussi la méthode BM basée sur la recherche harmonique (HS) [12]. Le problème majeur de la technique HS, c'est qu'elle ne fournit pas une balance effective entre les recherches globale et locale et elle stagne dans les dernières itérations [207].

Dans ce chapitre, un nouvel algorithme de BM basé sur une technique métaheuristique appelée la recherche fractale stochastique (SFS : stochastic fractal search) est proposé. Le choix de SFS est dû essentiellement au fait que cette technique peut fournir un bon équilibre entre l'exploitation et l'exploration et elle procure des bons résultats d'optimisation. L'algorithme développé est nommé SFS-BM, il utilise une implémentation parallèle pour calculer les déplacements de tous les blocs simultanément. Des nouvelles idées concernant l'initialisation, la fonction de fitness, la fenêtre de recherche et l'approximation de la fonction de fitness sont également proposées.

Dans la suite de ce chapitre, nous décrivons d'abord l'algorithme SFS et son implémentation parallèle. Puis, nous présentons notre algorithme SFS-BM. Ensuite, les performances de notre SFS-BM en termes de précision et de complexité de calcul sont évaluées avec six séquences vidéos et comparées avec d'autres algorithmes existants dans la littérature. Nous terminons ce chapitre par une conclusion.

III.2 La recherche fractale stochastique

Cette section est consacrée à l'algorithme SFS, nous commençons par décrire la technique SFS, ensuite, nous présentons l'implémentation parallèle utilisée.

III.2.1 Le principe de SFS

La recherche fractale stochastique (SFS) est une technique métaheuristique à base de population, elle appartient à la famille d'algorithmes évolutionnaires. Elle est proposée par Hamid Salimi en 2015 [158], et inspirée par le phénomène naturel de la croissance qui utilise le concept mathématique de fractal.
Le mot fractal est créé par Benoît Mandelbrot à partir de la racine latine *fractus*, qui signifie brisé, irrégulier. Une particularité des fractales comme de la nature est la répétition de formes similaires à différentes échelles d'observation. Ainsi, une partie d'un nuage ressemble au nuage tout entier, et un rocher rappel les formes de la montagne. Une forme typiquement fractale est celle du chou-fleur, ou du brocoli, dont les parties sont exactement à l'image du tout (voir figure. III.1). [1]

FIGURE III.1. : Exemples sur les objets fractales dans la nature

L'algorithme SFS commence par une initialisation aléatoire d'un nombre N_p de solutions candidates dans l'espace de recherche. Ensuite, cette population initiale est évaluée par une fonction de fitness pour mesurer la qualité de chaque solution et déterminer la meilleure solution. La technique SFS utilise par la suite trois processus principaux, un processus de diffusion et deux processus de mise à jour, pour converger itérativement vers la meilleure solution globale.

[1] http ://www.inexplique-endebat.com/article-fractales-l-univers-d-une-dimension-cachee-62731260.html

III.2.1.1 Le processus de diffusion

Dans le processus de diffusion, chaque solution candidate est diffusé et génère d'autres solutions. Pour réaliser ce processus, SFS utilise deux démarches aléatoires gaussiennes, définies par les équations.(III.1) et (III.2) en faisant une permutation aléatoire entre les deux. La figure. III.2 illustre un exemple de diffusion d'une solution.

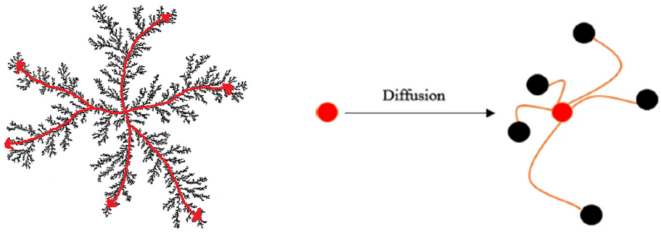


FIGURE III.2. : Le principe de la diffusion

$$x_i^g = Gaussian(\mu, \sigma) + (r G_{best} - r^{'} x_i) \tag{III.1}$$

$$x_i^g = Gaussian(\mu, \sigma) \tag{III.2}$$

$$\sigma = |\frac{\log(iter)}{iter}(x_i - G_{best})| \tag{III.3}$$

Où x_i est une solution candidate et G_{best} est la meilleure solution trouvée jusqu'à l'itération courante. Les deux nombres aléatoires r et $r^{'}$ sont tirés uniformément dans l'intervalle [0,1]. μ et σ représentent la moyenne et l'écart type des équations gaussiennes. La moyenne μ dans l'équation (III.1) est la meilleure solution globale G_{best}, mais dans l'équation (III.2) est la solution candidate x_i. L'écart type σ est calculé par l'équation (III.3) pour les deux gaussiennes.

La première équation gaussienne génère une nouvelle solution x_i^g dans le voisinage de la meilleure solution globale. Tandis que, la deuxième équation gaussienne génère une solution dans le voisinage de la solution candidate. La permutation aléatoire entre ces deux équations permet d'exploiter une fois la zone de la meilleure solution et une autre fois la zone de la solution candidate.

L'écart type σ, calculé par l'équation (III.3) prend une valeur croissante au début pour

que les solutions générées soient loin des solutions candidates, afin d'explorer différentes régions dans l'espace de recherche. Ensuite, σ prend une valeur décroissante pour concentrer la recherche dans les zones prometteuses. Figure III.3 illustre la variation des valeurs de σ.

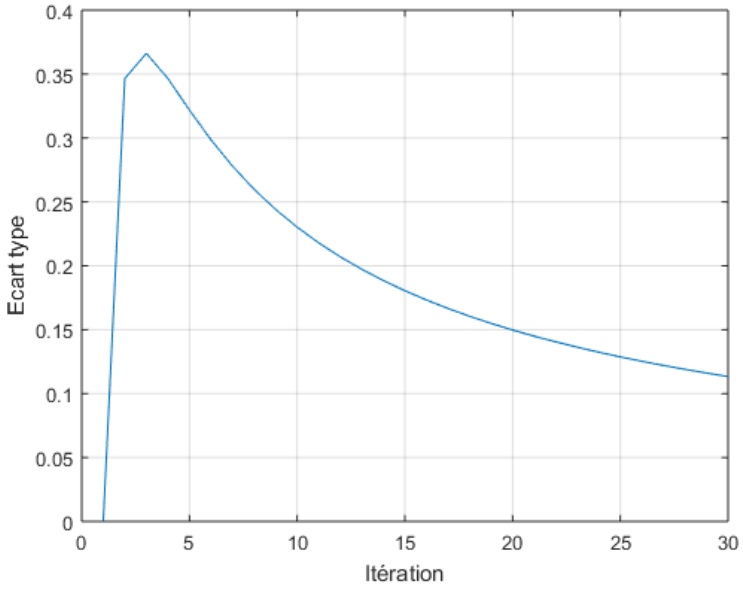


FIGURE III.3. : Valeurs de l'écart type σ en fonction du nombre d'itérations

III.2.1.2 Le premier processus de mise à jour

Dans le premier processus de mise à jour, une valeur de probabilité est donnée à chaque solution candidate à l'aide de l'équation (III.4), où $rank(x_i)$ représente le classement de la solution x_i. Ensuite, si une solution candidate a une valeur de probabilité inférieure à un nombre aléatoire uniforme r, ($Pa_i < r$), sa position est mise à jour par l'équation (III.5).

$$Pa_i = 1 - \frac{rank(x_i)}{N_p} \tag{III.4}$$

$$x_i^{'}(j) = x_r(j) - r \times (x_t(j) - x_i(j)) \tag{III.5}$$

Avec x_r et x_t sont deux solutions candidates sélectionnées aléatoirement de la population. Le but de ce processus est de changer les positions des mauvaises solutions et passer les meilleures solutions à l'itération suivante.

III.2.1.3 Le deuxième processus de mise à jour

Dans le deuxième processus de mise à jour, les valeurs de probabilités sont calculées de même manière comme dans le premier processus. Puis, chaque solution qui satisfait la condition suivante $Pa_i < r$, sa position est mise à jour par les équations (III.6) et (III.7) en faisant une permutation aléatoire entre eux.

$$x_i^{''} = x_i^{'} - r \times (x_t^{'} - G_{best}) \quad si\ r^{'} \leq 0.5 \tag{III.6}$$

$$x_i^{''} = x_i^{'} - r \times (x_t^{'} - x_r^{'}) \qquad si\ r^{'} > 0.5 \tag{III.7}$$

Où $x_r^{'}$ et $x_t^{'}$ sont deux solutions candidates sélectionnées à partir de la population.
Les processus de diffusion et de mise à jour sont répétés jusqu'à ce qu'un nombre d'itérations soit maximal.
L'algorithme 3 donne un pseudo-code pour SFS.

Initialisation de la taille de population N_p,
Initialisation du nombre d'itérations maximal,
Initialisation aléatoire des solutions candidates,
Pour iter **de** 1 **à** le nombre d'itérations maximal **faire**
  Évaluation des solutions candidates par la fonction de fitness
  Le processus de diffusion en utilisant les équations (III.1) et (III.2)
  Le premier processus de mise à jour en utilisant l'équation (III.5)
  Le deuxième processus de mise à jour en utilisant les équations (III.6) et (III.7)
Fin Pour

Algorithme 3 : L'algorithme SFS

La technique SFS est une technique évolutionnaire très efficace, elle peut fournir un bon équilibre entre l'exploitation et l'exploration. D'une part, le processus de diffusion qui génère de nouvelles solutions au voisinage de chaque solution permet l'exploitation des régions prometteuses, en plus l'écart type ou le rayon de recherche qui diminue au cours des itérations permet de balancer la recherche entre exploration et exploitation. D'autre part, les processus de mise à jour permettent de diversifier la recherche pour améliorer l'exploration.
L'auteur Hamidi [158] a comparé les résultats d'optimisation obtenus par la technique SFS avec ceux des autres métaheuristiques et il a démontré que cette méthode dépasse d'autres

méthodes dans le cas des fonctions de test uni-modales et multimodales. En plus, la technique SFS a été utilisée dans des applications diverses et les résultats ont été satisfaisants [208, 209, 210, 211].
L'inconvénient principal de la technique SFS, c'est que le nombre d'évaluations de la fonction de fitness est élevé.

III.2.2 Implémentation parallèle de SFS

En BM, pour calculer les vecteurs de déplacements de tous les blocs simultanément, nous proposons d'utiliser une implémentation parallèle de SFS. Au lieu d'une seule population, plusieurs sous-populations sont utilisées, chacune d'elle lui ai associée une fonction de fitness particulière. Les figures III.4 et III.5 présentent les organigrammes de SFS original et son l'implémentation parallèle.

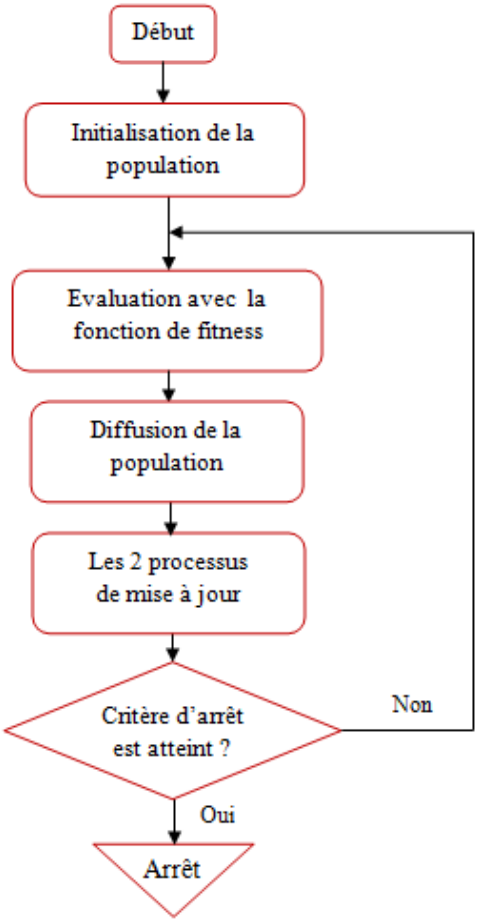


FIGURE III.4. : Organigramme de SFS avec une seule population

Contrairement à la version standard de SFS, l'implémentation parallèle III.5 initialise aléatoirement plusieurs sous-populations dans des espaces de recherche différents. Chaque sous-population est désignée pour trouver un optimum différent, c'est pour cela, chacune d'elles est évaluée avec une fonction de fitness particulière. Les sous-populations sont ensuite mises à jour avec les trois processus de SFS. Ces dernières se répètent jusqu'à ce que le critère d'arrêt soit atteint (secondaire). L'algorithme est doté d'une condition d'arrêt qui termine le fonctionnement de toutes les sous-populations (critère d'arrêt principal).

III.3 L'algorithme SFS-BM

Dans l'algorithme SFS-BM, chaque deux images successives de la séquence vidéo (courante et référence) sont divisées en blocs. Puis, pour chaque bloc dans l'image courante, une sous-population est initialisée dans la fenêtre de recherche correspondante. Les processus de l'algorithme SFS sont ensuite exécutés pour plusieurs itérations. Les étapes de l'algorithme SFS-BM sont détaillées comme suit :

III.3.1 Initialisation

Notre algorithme SFS-BM calcule les vecteurs des déplacements de tous les blocs simultanément, le modèle utilisé pour l'initialisation peut être décrit par la matrice (III.8).

$$\begin{bmatrix} R_{k,s} & R_{k,s+1} & R_{k,s+2} & \cdots & R_{k,S} \\ R_{k+1,s} & R_{k+1,s+1} & R_{k+1,s+2} & \cdots & R_{k+1,S} \\ R_{k+2,s} & R_{k+2,s+1} & R_{k+2,s+2} & \cdots & R_{k+2,S} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ R_{K,s} & R_{K,s+1} & R_{K,s+2} & \cdots & R_{K,S} \end{bmatrix} \tag{III.8}$$

$R_{k,s}$ indique le bloc candidat numéro s dans l'image de référence, $s = 1,2,\cdots Np$, pour le bloc numéro k, $k = 1,2,\cdots K$.
Où N_p est la taille de chaque sous-population et $K = \frac{M}{B_x} \times \frac{N}{B_y}$ représente le nombre maximal de blocs dans l'image. Avec M et N sont les tailles de l'image.

D'après les travaux de [212] et [213], nous pouvons améliorer considérablement les performances des techniques métaheuristiques, si la population initiale de solutions candidates est générée selon des connaissances sur le domaine (c-à-d des solutions non-aléatoires).

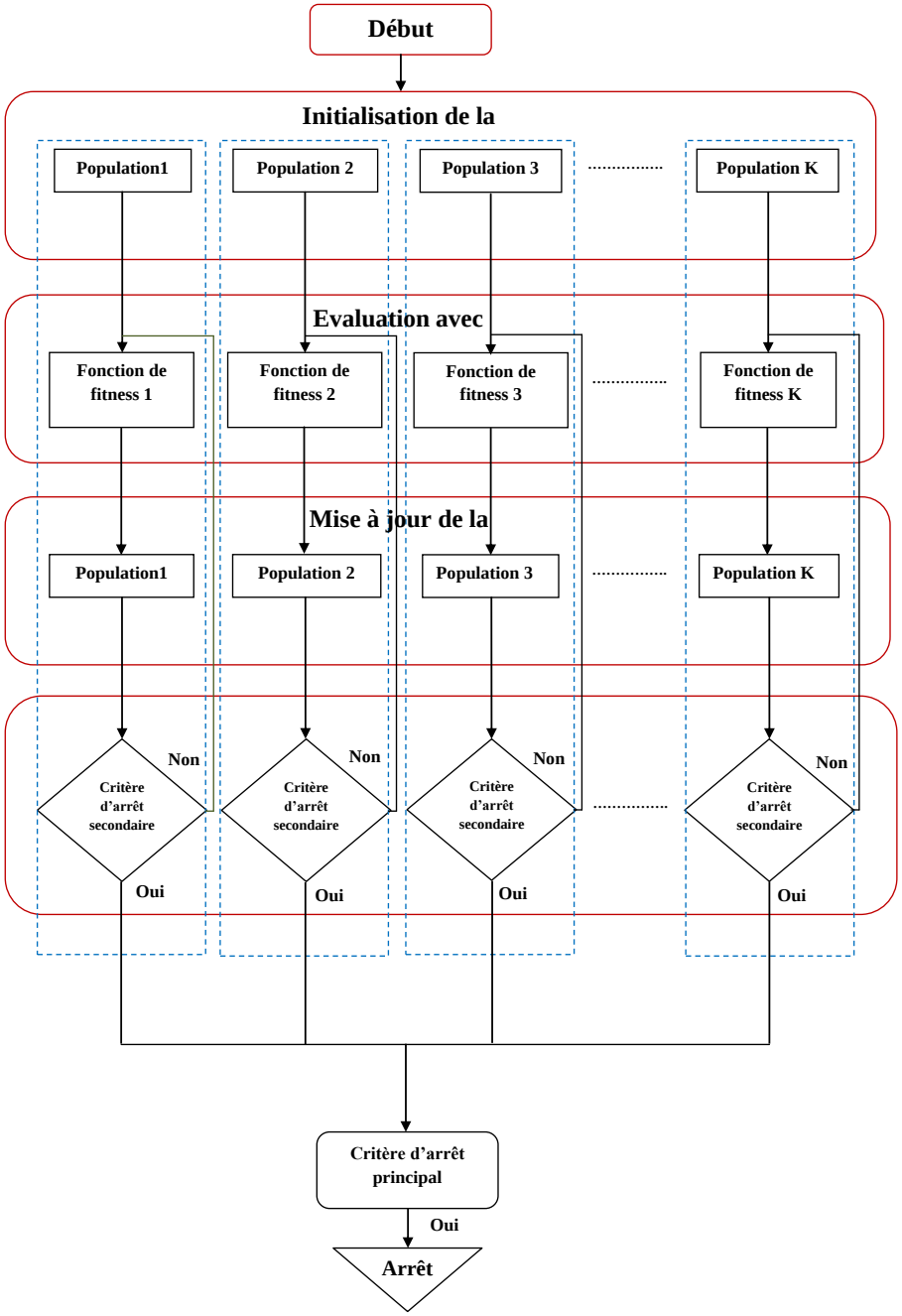


FIGURE III.5. : Organigramme de SFS avec une implémentation parallèle

L'étude statistique menée par [20], sur la distribution des vecteurs de mouvement dans le problème de BM, a démontré que les blocs les plus similaires se trouvent dans le centre de la fenêtre de recherche avec une probabilité de 98% pour les séquences de faible mouvement, et 53.5% pour les séquences de mouvement médian et 36.9% pour les séquences de mouvement rapide. Il ont déclaré aussi que l'utilisation d'un modèle fixe pour initialiser les blocs candidats améliore l'efficacité de l'algorithme BM.

Contrairement à l'initialisation aléatoire utilisée généralement dans les algorithmes évolutionnaires, dans notre SFS-BM les blocs candidats sont sélectionnés soigneusement afin d'accélérer la convergence. Vingt blocs candidats sont sélectionnés comme suit :

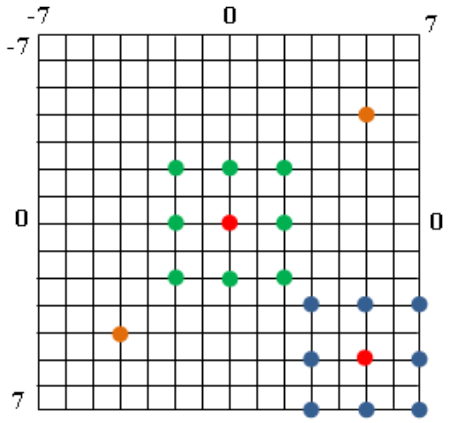


FIGURE III.6. : Nouvelle stratégie d'initialisation

- Neuf blocs candidats, le point rouge et les huit points verts qui l'entourent montrés dans la figure.III.6, sont situés dans les mêmes positions que le bloc courant et ses voisins, et correspondent aux vecteurs de mouvement suivants {(0,0) ; (0,-1) ;(0,1) ; (-1,-1) ; (-1,0) ; (-1,1) ; (1,-1) ; (1,0) ; (1,1)}. Cette sélection est basée sur l'hypothèse que le bloc courant se déplace lentement.

- Neuf blocs candidats, le point rouge et les points bleus qui l'entourent, sont situés dans les mêmes positions que celles du bloc prédit et de ses voisins. Avec l'équation (III.9) on calcule les coordonnées du bloc prédit en utilisant les vecteurs de mouvement trouvés dans l'image précédente.

$$B_{predit} = B + MV \tag{III.9}$$

  Il est à noter que la prédiction est plus précise si la vitesse de déplacement du bloc est

constante. Pour obtenir une prédiction robuste, notre algorithme SFS-BM commence à partir de la deuxième image de la séquence vidéo et les vecteurs de mouvement dans la première image sont calculés à l'aide de l'algorithme FSA.

- Deux blocs candidats, les points en orange, sont sélectionnés loin des autres afin d'explorer des zones différentes dans la fenêtre de recherche et éviter les optima locaux.

III.3.2 La fonction de fitness

Pour mesurer la similarité ou la dis-similarité entre deux images différentes, nombreux critères ont été formulés, chaque critère a ses propres avantages et inconvénients [214, 215]. Parmi ces critères, il y a :

La moyenne des différences carrées (MSE : mean square error) mesurée comme suit :

$$MSE = \frac{\sum_{i=1}^{B_x} \sum_{j=1}^{B_y} (C_{i,j} - R_{i,j})^2}{B_x \times B_y} \tag{III.10}$$

C et R représentent le bloc courant et le bloc candidat de l'image référence, B_x et B_y sont les tailles du bloc.
Les avantages du critère MSE sont :

- Le calcul du MSE est simple ;
- MSE représente la distance euclidienne entre deux choses, et prend en compte la perception humaine, il est également utilisé pour le calcul du $PSNR$;
- La fonction MSE est très utilisée en optimisation, puisque elle satisfait les propriétés de convexité, symétrie et la différentiabilité, donc les calculs du gradient et de la matrice hessienne sont faciles ;
- C'est le critère le plus utilisé dans la littérature, les chercheurs utilisent aussi le MSE pour la comparaison entre les performances de différents algorithmes.

La norme L1, dite aussi norme Manhattan ou la somme des différences absolues (*SAD*), est définie comme suit :

$$SAD = \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} |C_{i,j} - R_{i,j}| \tag{III.11}$$

Les avantages du critère *SAD* sont :

- Le calcul de *SAD* est extrêmement facile ;
- Il peut produire des résultats de dis-similarité aussi exacts que ceux produits par des mesures plus coûteuses.

Nous proposons de combiner ces deux critères : *SAD* et *MSE* afin de bénéficier de leurs points forts. L'idée de combiner plusieurs critères de similarité est largement utilisée dans le domaine d'estimation de mouvement, en particulier avec les algorithmes de flot optique [91, 62], mais elle n'a pas été utilisée avec les algorithmes de mise en correspondance de blocs. Nous attirons également votre attention sur le fait que l'optimisation d'une fonction de fitness complexe à l'aide des techniques métaheuristiques est plus facile et plus simple qu'avec les autres méthodes exactes. La fonction de fitness proposée est formulée par l'équation (III.12) :

$$\boldsymbol{F} = \frac{1}{(B_x \times B_y)} \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} (1-a) \times E^2 + a \times |E| \tag{III.12}$$

Où

$$E = C_{i,j} - R_{i,j} \tag{III.13}$$

a est un paramètre de pondération fixé à 0,5.

Il est à noter que chaque sous-population a une fonction de fitness différente, cela dépend du bloc C dans l'image courante et les blocs candidats dans la fenêtre de recherche correspondante. En plus, il n'y a pas une coopération entre les sous-populations.

III.3.3 Les processus de SFS

Après l'initialisation et l'évaluation, les processus de SFS (diffusion et mise à jour) sont exécutés.

III.3.4 La fenêtre de recherche adaptative

Dans la section I.3.2.3.e, nous avons souligné que le règlement de la taille de la fenêtre de recherche influence la précision d'estimation de mouvement et ainsi la complexité de calcul. Nous proposons dans notre algorithme SFS-BM l'utilisation des tailles adaptatives de la fenêtre de recherche, ça dépend le mouvement estimé dans la séquence vidéo. Pour chaque image dans la séquence vidéo et pour chaque bloc dans l'image, on utilise les équations (III.14) et (III.15) pour déterminer les tailles horizontale et verticale de la fenêtre de recherche. Elles sont basées sur le vecteur de mouvement prédit du bloc et le mouvement maximal estimé dans les images précédentes.

$$W^h_{k,t} = max(|MV^h|) + |MVP^h_{k,t}| \tag{III.14}$$

$$W^v_{k,t} = max(|MV^v|) + |MVP^v_{k,t}| \tag{III.15}$$

Où t représente l'indice de l'image courante, MV^h et MV^v sont les vecteurs de mouvement horizontaux et verticaux trouvés dans les images précédentes. $MVP^h_{k,t}$ et $MVP^v_{k,t}$ sont les vecteurs de mouvement prédits du bloc k.

Les vecteurs de mouvement prédits du bloc k peuvent être suffisants pour déterminer les nouvelles tailles de la fenêtre, mais comme indiqué précédemment, le vecteur de mouvement prédit d'un bloc est précis seulement si sa vitesse est constante. Pour prendre en considération le cas d'une augmentation soudaine de la vitesse, nous avons ajouté la valeur maximale du mouvement estimé dans la séquence. Pour assurer que les tailles de la fenêtre ne sont pas supérieures qu'une taille maximale W_{max} et non inférieure à la taille minimale $W_{min} = 1$, nous avons utilisé les équations (III.16) et (III.17).

$$W^h_{k,t} = max(min(W^h_{k,t}, W_{max}), W_{min}) \tag{III.16}$$

$$W^v_{k,t} = max(min(W^v_{k,t}, W_{max}), W_{min}) \tag{III.17}$$

III.3.5 L'approximation de la fonction de fitness modifiée

Pour réduire le nombre d'évaluations de la fonction de fitness, **la méthode d'approximation de la fonction de fitness** [20] a été proposée. Elle évalue quelques blocs à travers la fonction de fitness réelle, et estime seulement les autres.
Soient les deux blocs candidats : P_n qui n'est pas encore évalué et P_v un bloc évalué mais sa

valeur de fitness n'est pas la meilleure. Si la distance entre ces deux blocs est inférieure à une distance d, la valeur de fitness du bloc P_n est estimée en lui affectant la même valeur de fitness du bloc P_v, Autrement le bloc est évalué. L'algorithme 4 représente un pseudo-code pour cette méthode.

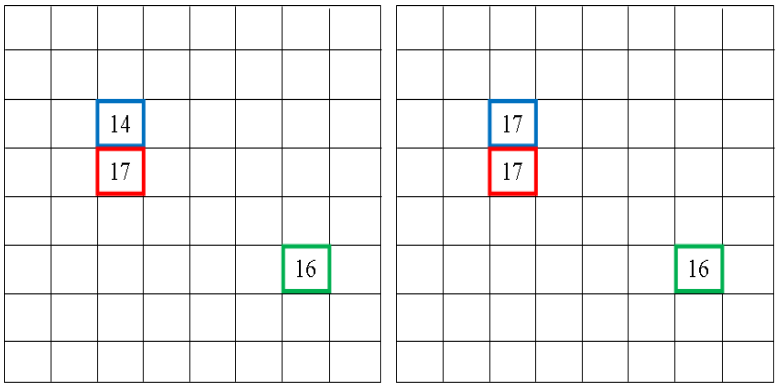
Si (La distance entre P_n et P_v est inférieure à une distance d) **Alors**
  Si (P_v n'est pas le meilleur bloc trouvé jusqu'à l'itération courante) **Alors**
    $F_n = F_v$
  Sinon
    P_n est évalué à travers la fonction de fitness réelle
  Fin Si
Sinon
  P_n est évalué à travers la fonction de fitness réelle ;
Fin Si

Algorithme 4 : L'approximation de la fonction de fitness

L'algorithme d'élimination successive (SEA) est une autre méthode proposée dans la littérature pour réduire la complexité de calcul. Comme il a été expliqué dans la section I.3.2.3.b, la méthode SEA évalue seulement les blocs candidats qui satisfont la contrainte (III.18) [16].

$$S_C - SAD_{min} \leq S_R \leq S_C + SAD_{min} \tag{III.18}$$

Revenant à la méthode d'approximation de la fonction de fitness [20]. Si un nouveau bloc candidat P_n est meilleur que le bloc voisin P_v, l'algorithme peut stagner dans un optimum local. La figure III.7 donne un exemple sur les valeurs de fitness réelles et approximées par la méthode [20]. On remarque que, la valeur de fitness réelle du bloc en bleu (P_n) est 14, donc elle meilleure que le bloc voisin en rouge (P_v) et le bloc en vert (le meilleur bloc trouvé). Mais, en appliquant la méthode d'approximation, il prend la valeur 17, donc malgré que le bloc en bleu est meilleur que le bloc en vert, mais ce dernier reste le meilleur bloc trouvé. Pour cela, nous avons proposé de remplacer la condition qui décide si un bloc candidat doit être évalué ou estimé par la condition de l'algorithme d'élimination successive.
L'approximation de la fonction de fitness modifiée est présentée dans de l'algorithme 5.



(a) Les valeurs de fitness réelles (b) Les valeurs de fitness approximées

FIGURE III.7. : Exemple sur les valeurs de fitness

Si (La distance entre P_n et P_v est inférieure à une distance d) **Alors**
  Si ($S_C - SAD_{min} \leq S_R \leq S_C + SAD_{min}$) **Alors**
    P_n est évalué à travers la fonction de fitness réelle
  Sinon
    $F_n = F_v$
  Fin Si
Sinon
  P_n est évalué à travers la fonction de fitness réelle ;
Fin Si

Algorithme 5 : L'approximation de la fonction de fitness modifiée

III.3.6 Réduction de la complexité de calcul

Pour réduire encore plus la complexité du calcul, deux stratégies populaires sont intégrées dans notre algorithme : l'exploitation des valeurs de la fonction de fitness et le préjugement du mouvement zéro.

III.3.6.1 L'exploitation des valeurs de la fonction de fitness

Cette méthode empêche les algorithmes métaheuristiques de la réévaluation des mêmes solutions plus d'une fois. Il attribue un indicateur à chaque solution candidate possible. L'indicateur d'une solution qui n'est pas encore évaluée prend la valeur '0', celui d'une solution

déjà évaluée prend la valeur '1' [11]. Les valeurs de fitness calculées sont sauvegardées, pour les utiliser si l'algorithme revient à une solution déjà évaluée.

III.3.6.2 Le pré-jugement du mouvement zéro

Cette approche détermine les blocs statiques qui n'ont pas besoin d'une recherche pour les meilleurs blocs correspondants ; leurs vecteurs de mouvement sont $(0,0)$. Elle mesure la disparité entre le bloc dans l'image courante et le bloc candidat localisé dans la même position dans l'image référence. Si cette mesure est inférieure d'un seuil prédéterminé, le bloc dans l'image courante est considéré comme un bloc statique [132].
La valeur du seuil choisi est très importante, car c'est elle qui décide si un bloc est statique ou non. Elle doit être sélectionnée à partir de l'intervalle des valeurs possibles de disparité. Nous avons remarqué que les valeurs de disparité se trouvent dans les intervalles suivants :

- $[0,43-16]$ si le mouvement dans la séquence vidéo est faible,
- $[0,6-76]$ si le mouvement dans la séquence vidéo est médian,
- $[0.75-1358]$ si le mouvement dans la séquence vidéo est rapide.

La valeur de ce seuil ne doit pas être trop grande pour ne pas considérer un bloc comme statique alors qu'il est en fait en mouvement. Et elle ne doit pas être très faible pour ne pas considérer un bloc en mouvement alors qu'il est en fait statique ; un temps de calcul sera inutilement dépensé dans la recherche du vecteur de mouvement non-existant. Pour éviter ce problème, nous avons choisi une valeur médiane adaptée à tous les types de mouvements qui est $seuil = 1,5$.

À la fin de chaque itération, les trois mauvaises solutions dans chaque sous-population sont rejetées pour réduire de plus en plus la complexité du calcul.

III.3.7 Critère d'arrêt

Pour converger vers le bloc le plus similaire, les processus sont répétés jusqu'à ce qu'un critère d'arrêt soit atteint. Puisque notre algorithme calcule les vecteurs de mouvement de tous les blocs simultanément, mais indépendamment, il n'y a pas une coopération entre eux durant

la recherche, nous avons défini deux critères d'arrêt.

- Le critère principal : les processus sont arrêtés si tous les blocs similaires sont trouvés ou le nombre maximal d'itérations est atteint.

- Le critère secondaire : pour chaque bloc, la recherche du bloc le plus similaire s'arrête si la meilleure valeur de fitness trouvée est inférieure à une valeur du seuil. Celui-ci est fixé à 1,5 comme le pré-jugement du mouvement zéro.

Nous avons fixé, empiriquement, le nombre maximal d'itérations à 3 itérations. L'algorithme 6 donne un pseudo-code de l'algorithme SFS-BM.

Initialisation de la taille de chaque sous-population N_p,
Initialisation du nombre maximal d'itérations,
Initialisation de la taille de la fenêtre de recherche w_{max},
Initialisation des tailles de bloc $B_x \times B_y$,
Lire la séquence vidéo I,
Pour t **de** 1 **à** le nombre maximal des images **faire**
  Image courante = $I(t)$
  Image référence = $I(t-1)$ ou $I(t+1)$
  Diviser l'image courante en blocs
  Déterminer les blocs statiques
  Si (t=1) **Alors**
    Calculer les vecteurs de mouvements avec la technique FSA
  Sinon
    Initialisation des blocs candidats comme montré dans la figure. III.6
    Pour *iter* **de** 1 **à** le nombre maximal d'itérations **faire**
      Évaluation des blocs candidats par la fonction de fitness (III.12) ou l'estimation en utilisant algorithme 5
      Le processus de diffusion en utilisant les équations. (III.1) et (III.2)
      Le premier processus de mise à jour en utilisant l'équation. (III.5)
      Le deuxième processus de mise à jour en utilisant les équations. (III.6) et (III.7)
    Fin Pour
    Les vecteurs de mouvements sont calculés en utilisant les meilleurs blocs trouvés.
  Fin Si
  Déterminer le mouvement prédit pour chaque bloc en utilisant l'équation. (III.9)
  Mise à jour des tailles de la fenêtre de recherche en utilisant les équations. (III.14), (III.15), (III.16) et (III.17)
Fin Pour

Algorithme 6 : Algorithme SFS-BM

III.4 Résultats et discussion

À travers nos simulations, nous avons calculé les vecteurs de mouvement dans six séquences vidéo avec différents formats et types de mouvement. Les caractéristiques des séquences vidéo utilisées sont présentées dans le tableau III.1 et une image de chaque séquence est illustrée sur la figure III.8.

TABLE III.1. : Les séquences vidéos utilisées et leurs caractéristiques

La Séquence	Format	Taille de l'image	Type de mouvement	Nombre total des images	Taille maximale de la fenêtre
Akiyo	Qcif	(144×176)	Faible	300	8
Container	Qcif	(144×176)	Faible	300	8
Carphone	Qcif	(144×176)	Médian	382	8
Foreman	Qcif	(144×176)	Médian	300	8
Hall	Cif	(288×352)	Rapide	300	8
Stefan	Cif	(288×352)	Rapide	90	16

(a) Akiyo (b) Container (c) Carphone

(d) Foreman (e) Hall (f) Stefan

FIGURE III.8. : Les séquences vidéos

Le tableau III.2 récapitule les valeurs des paramètres utilisés :

TABLE III.2. : Les paramètres de l'algorithmes SFS-BM

Paramètre	Description	Valeur
d	La distance de l'approximation de fitness	3
$seuil$	Le seuil du préjugement zéro	1.5
$Iter_{max}$	Nombre d'itération maximal	3
W_{max}	Taille maximale de la fenêtre	Selon la séquence vidéo (définie dans le tableau III.1)
W_{min}	Taille minimale de la fenêtre	1

Les performances de chaque algorithme BM dépendent de l'aptitude de l'algorithme à réduire la complexité de calcul de FSA, et assurer au même temps que la dégradation au niveau de *PSNR* ne sera pas gênante. Donc, des valeurs minimales de D_{PSNR} et du nombre de blocs évalués *NB*. Le tableau III.3 présente les résultats de l'algorithme FSA, en terme de *PSNR* et *NB*.

TABLE III.3. : Les résultats de l'algorithme FSA

FSA	Akiyo	Container	Carphone	Foreman	Hall	Stefan
PSNR	43.18	44.29	31.51	31.69	25.95	34.81
NB	236,63	236,63	236,63	236,63	262,62	984,91

Le reste de cette section est divisé en deux parties. Dans la première partie, nous analysons l'impact de chaque idée proposée sur les performances de l'algorithme SFS-BM. Dans la deuxième partie, nous comparons les performances de l'algorithme SFS-BM avec d'autres algorithmes BM connus dans la littérature.

III.4.1 Analyse de l'algorithme

Dans cette partie, les quatre nouvelles idées proposées dans SFS-BM concernant l'initialisation, la fonction de fitness, la fenêtre de recherche et l'approximation de la fonction de fitness, sont analysées. Les pourcentages d'amélioration remportés par chacune d'elles en *PSNR* et en nombre de blocs évalué *NB* sont présentés dans le tableau.III.4.
Le pourcentage d'amélioration d'un algorithme ALG1 par rapport à l'algorithme ALG2 est

calculé comme suit :

$$Amelioration_{PSNR} = -(\frac{PSNR_{ALG2} - PSNR_{ALG1}}{PSNR_{ALG2}}) \times 100\% \quad \text{(III.19)}$$

$$Amelioration_{NB} = (\frac{NB_{ALG2} - NB_{ALG1}}{NB_{ALG2}}) \times 100\% \quad \text{(III.20)}$$

TABLE III.4. : Les pourcentage d'amélioration de chaque idée

Amélioration en terme de	Nouvelle Initialisation	Fonction de fitness combinée	Fenêtre adaptative	Approximation de fitness [20]	Approximation de fitness modifiés
PSNR %	34.51	0.15	7.85	-5.21	-0.05
NB %	51.32	0.08	20.44	56.61	8.46

Les pourcentages d'amélioration de chaque idée sont présentés dans le tableau.III.4. À partir de ces résultats nous remarquons que :

- L'initialisation proposée a remporté des pourcentages d'amélioration significatives plus que 34 % pour le *PSNR* et 51 % pour *NB*. On peut justifier l'efficacité du modèle d'initialisation proposé par le fait que les blocs initialisés sont distribués d'une façon que différentes zones de la fenêtre sont explorées et les zones prometteuses sont bien exploitées.

- La fonction de fitness proposée a amélioré les résultats en *PSNR* et *NB* avec des pourcentages d'améliorations égales à 0,15 % et 0,08 % respectivement. Ce qui prouve que la combinaison de deux critères de similarité peut effectivement améliorer les résultats.

- La fenêtre de recherche adaptative proposée donne un pourcentage d'amélioration en *NB* dépasse 20 %. Malgré que le but derrière l'utilisation d'une fenêtre adaptative, c'est diminuer le nombre de blocs évalués, elle a aussi amélioré le *PSNR*. On peut justifier ce comportement comme suit : généralement, les blocs candidats initiales sont distribués dans toute la fenêtre de recherche. Si ces blocs sont très loin de l'optimum global, l'algorithme peut tomber dans un optimum local. Mais si la fenêtre de recherche est bien réglé selon le type de mouvement existant dans la séquence vidéo, les blocs candidats sont initialisés autour de l'optimum global et les résultats s'améliorent.

- L'utilisation de la méthode d'approximation de la fonction de fitness [20] a amélioré la complexité de calcul avec un pourcentage de 56.61%, mais a dégradé le *PSNR* avec un pourcentage de -5.21 %.

- La méthode d'approximation de la fonction de fitness modifiée, que nous avons proposé, n'a pas amélioré la complexité de calcul avec le même pourcentage que la méthode originale, mais elle a assuré que la dégradation en *PSNR* est minimale -0.05%.

De cette analyse, on peut conclure que les idées proposées peuvent améliorer les résultats en précision *PSNR* aussi que la complexité de calcul *NB*.

III.4.2 Comparaison avec d'autre méthodes BM

Les résultats obtenus par notre SFS-BM sont comparés avec d'autres algorithmes BM connus dans la littérature qui sont :

- La recherche exhaustive (FSA) [8] (1981).

- Sept algorithmes BM avec recherche orientée : La recherche en trois étapes (TSS) [9] (1994), le nouveau TSS (NTSS) [139] (1994), le TSS simple et efficace (SES) [216] (1996), la recherche en quatre étapes (4SS) [120] (1997), la recherche en diamant (DS) [15] (2000), la prédiction moyenne pour BM (MPBM) [217] (2011) et la recherche star diamant (SDDS) [218] (2017).

- Six algorithmes BM basés sur les métaheuristiques : PSO-BM [172] (1995), BH-BM [177] (2013), SCA-BM [161] (2014), GWO-BM [176] (2016), SSA-BM [241] (2017) et MS-BM [219] (2018).

- Deux algorithmes de BM basés sur l'apprentissage d'automates, PLA-BM et TPLA-BM [220] (2016).

En terme de D_{psnr}, les résultats présentés dans le tableau III.5 indiquent que :

- Pour les séquences vidéos avec des mouvements faibles, *Akiyo* et *Container*, notre algorithme SFS-BM et tous les autres algorithmes BM obtiennent des valeurs de D_{psnr} très

TABLE III.5. : Les valeurs de D_{psnr} obtenues avec les différents algorithmes BM

Algorithme	Akiyo	Container	Carphone	Foreman	Hall	Stefan	Moyenne	Classement
FSA	0	0	0	0	0	0	0	1
TSS	0	-0,2	-3,92	-7,32	-0,14	-18,52	-5,01	16
NTSS	0	-0,15	-3,67	-3,57	-0,11	-13,2	-3,45	9
4SS	0	-0,15	-4,01	-7,44	-0,14	-17,48	-4,87	13
SES	0	-0,15	-4,5	-7,73	-0,25	-17,64	-5,04	17
DS	0	-0,13	-0,79	-1,59	-0,14	-3,73	-1,06	7
MPBM	-0.14	0.01	-0.77	-0.68	-0.15	-1.58	-0.55	3
SDDS	-0.08	0.01	-0.62	-1.14	-0.11	-3.16	-0.85	5
PSO-BM	-0,25	0,01	-1,58	-4,28	-0,32	-10,88	-2.88	8
BH-BM	-0,11	-0,52	-3,05	-6,12	-1,20	-17,42	-4.74	12
GWO-BM	-0,10	-0,37	-3,02	-5,78	-1,11	-17,09	-4.58	11
SCA-BM	-0,09	-0,72	-3,44	-6,09	-1,35	-18,04	-4.95	15
SSA-BM	-0,12	-0,58	-2,90	-5,49	-1,10	-15,86	-4.34	10
MS-BM	-0,12	-0,64	-3,28	-6,25	-1,31	-17,76	-4.89	14
PLA-BM	0	0	-0,28	-0,82	//	-3,19	-0,85	6
TPLA-BM	0	0	-0,28	-0,85	//	-2,92	-0,81	4
SFS-BM	-0,02	0	0	0	-0,02	-2,04	**-0,34**	2

petites.

- Pour les séquences vidéos avec des mouvements médians, *Carphone* et *Foreman*, on remarque l'augmentation des valeurs de D_{psnr} dans tous les algorithmes, surtout les algorithmes avec une recherche orientée comme TSS, 4SS, etc. Par contre, notre algorithme obtient des valeurs de D_{psnr} égales à zéro.

- Pour les séquences vidéos avec des mouvements rapides, *Hall* et *Stefan*, les méthodes de BM basées sur les métaheuristiques comme PSO-BM, SSA-BM, GWO-BM et SFS-BM sont meilleures que les méthodes de recherche orientée comme TSS, 4SS et SES, puisque la nature stochastique des techniques métaheuristiques permet d'éviter les optima locaux mieux que les autres techniques.

D'après ces résultats, la valeur moyenne de D_{psnr} obtenue avec notre algorithme SFS-BM est la plus petite par rapport aux autres méthodes utilisées pour la comparaison.

En terme du nombre moyen de blocs évalués *NB*, le tableau.III.6 indique que :

- Pour les séquences *Akiyo* et *Container* (mouvement faible), le nombre de blocs évalués

TABLE III.6. : Le nombre de blocs évalués par les différents algorithmes BM

Algorithme	Akiyo	Container	Carphone	Foreman	Hall	Stefan	Moyenne	Classement
FSA	236,63	236,63	236,63	236,63	262,62	984,91	365,67	17
TSS	21,48	25	25	25	23,25	25	24,12	15
NTSS	14,68	17,2	21,8	22,1	16,92	25,4	19,68	13
SES	16,2	19,2	18,86	18,85	16,92	17,45	17,91	11
4SS	14,65	19	25,25	24,8	16,25	25,3	20,87	14
DS	11,43	7,5	12,5	13,4	12,89	15,2	12,15	5
MPBM	10.04	5.91	11.22	11.49	11.06	11.64	10.23	2
SDDS	11.45	11.42	13.06	14.83	12.88	17.25	13.48	6
PSO-BM	12,60	6,11	13,32	13,72	12,02	14,77	12.09	4
BH-BM	15,18	8,04	15,20	15,25	15,22	16,25	14.19	7
SCA-BM	19,79	10,42	19,79	19,79	20,01	20,65	18.41	12
GWO-BM	15,65	8,40	15,60	15,66	15,95	16,55	14.64	8
SSA-BM	16,61	8,79	16,77	17,02	17,04	19,20	15.90	10
MS-BM	16,04	8,53	16,02	16,03	16,16	16,58	14.89	9
PLA-BM	49,7	75,27	71,54	77,5	//	131,7	81,14	16
TPLA-BM	8,69	10,85	9,17	10,09	//	13,9	10,54	3
SFS-BM	1,41	1,41	7,52	14,62	15,45	19,65	**10,01**	1

par notre algorithme SFS-BM est petit par rapport les autres méthodes BM.

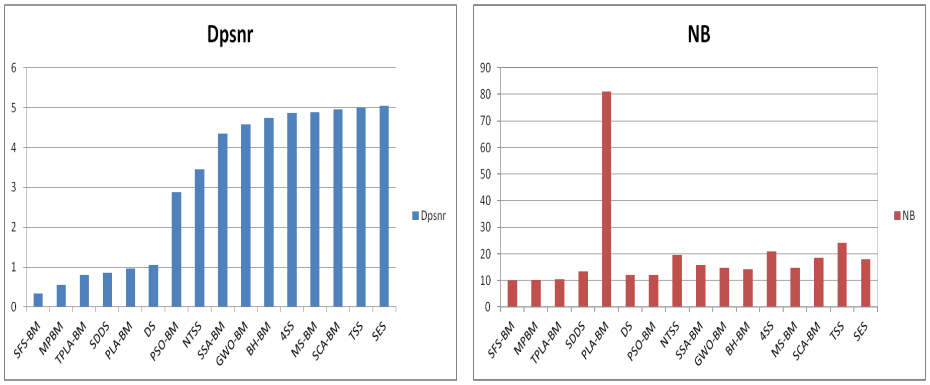
- Pour les séquences *Carphone* et *Foreman* (mouvement médian), on remarque une augmentation dans le nombre *NB*. La fenêtre adaptative utilisée en SFS-BM permet d'augmenter la taille de la fenêtre de recherche avec l'augmentation de l'amplitude du mouvement, ce qui fait le nombre de blocs évalués augmente de même.

- Pour les séquences *Hall* et *Stefan* (mouvement rapide), le nombre de blocs évalués par SFS-BM est élevé, cela revient à la fenêtre de recherche adaptative utilisée qui dépend au mouvement maximal estimé. Donc, la fenêtre de recherche adaptative proposée réduit le nombre *NB* seulement qu'avec les séquences vidéos de mouvement faible et médian.

D'après ces résultats, le nombre moyen des blocs évalués par notre SFS-BM est le plus petit par rapport aux autres méthodes BM citées. Ce qui confirme que la convergence de notre SFS-BM vers le minimum global est plus rapide que les autres méthodes.

Le temps de calcul des algorithmes BM est proportionnel aux nombres de blocs évalués *NB*. Mais, il faut noter ici que, les processus utilisés par notre méthode SFS-BM pour trouver les vecteurs de mouvement prédits et le calcul de la fenêtre adaptative augmentent aussi le

temps de calcul. Cependant, les stratégies de pré-jugement du mouvement zéro et d'approximation de la valeur de fitness peuvent réduire le temps de calcul. Notre méthode SFS-BM est meilleure que la méthode FSA, en temps de calcul, avec un pourcentage d'amélioration égale à 82.45%.



(a) D_{psnr} (b) NB

FIGURE III.9. : Graphe sur les valeurs de D_{psnr} et NB obtenues avec les différents algorithmes BM

Les graphes dans la figure III.9 présentent les résultats obtenus par notre méthode SFS-BM et les autres méthodes BM, en termes de D_{psnr} et NB. Il est clair que SFS-BM a la plus petite valeur de D_{psnr} et de NB par rapport aux autres méthodes citées.

TABLE III.7. : L'amélioration de SFS-BM par rapport aux autres algorithmes BM

Amélioration en terme de	SFS-BM / DS	SFS-BM / PSO-BM	SFS-BM / TPLA-BM
PSNR %	67.92	88.19	58.02
NB %	17.61	17.20	5.02

Dans le tableau III.7, nous présentons les pourcentages d'amélioration de SFS-BM par rapport aux algorithmes : DS, PSO-BM et TPLA-BM, en terme de précision et complexité de calcul. Nous pouvons résumer les résultats comme suit :

- Les algorithmes de BM avec recherche orientée comme DS se basent sur un modèle fixe pour explorer initialement le voisinage direct du bloc puis ils suivent la direction de la ressemblance croissante pour trouver le bloc le plus similaire. Notre algorithme SFS-BM utilise également un modèle fixe pour sélectionner la population initiale. Mais,

contrairement à DS, SFS-BM utilise ensuite des processus aléatoires pour explorer différentes régions dans la fenêtre de recherche. La nature stochastique de SFS-BM a permis d'éviter les optima locaux et donner des pourcentages d'amélioration significatives égales à 67.92% et 17.61% en terme de précision et complexité de calcul.

- Les algorithmes PSO-BM et SFS-BM sont deux techniques BM basées sur les métaheuristiques. Notre SFS-BM est meilleure que PSO-BM avec les pourcentages suivants : 88.19% et 17.20%, ce qui confirme la supériorité des algorithmes évolutionnaires et spécialement la technique SFS.

- La technique TPLA-BM est aussi de nature stochastique, puisqu'elle sélectionne à chaque instant une nouvelle action aléatoirement. La différence entre cette méthode et les métaheuristiques, c'est que ces dernières sont basées sur des modèles mathématiques inspirés par des phénomènes naturels intéressants capables à combiner entre l'exploration et l'exploitation. Notre technique SFS-BM est meilleure que TPLA-BM avec les pourcentages : 58.02% et 5.02 %.

Pour conclure, nous pouvons constater que notre méthode SFS-BM est capable d'obtenir des résultats satisfaisants, en précision et en complexité de calcul, meilleurs que plusieurs d'autres algorithmes connus dans la littérature.

III.5 Conclusion

Dans ce chapitre, nous avons présenté un nouvel algorithme de BM basé sur la technique SFS nommé SFS-BM et en utilisant une implémentation parallèle pour calculer les vecteurs de mouvement de tous les blocs simultanément. Quatre nouvelles idées sont ajoutées afin d'améliorer les résultats. À partir de l'analyse faite sur chaque idée, nous avons confirmé l'amélioration remportée par chaque idée. L'algorithme SFS-BM a été comparé avec d'autres algorithmes BM tels que TSS, 4SS, DS, PSO-BM et SSA-BM, etc. Les résultats prouvent que notre algorithme SFS-BM dépasse pour les scénarios étudiés tous les algorithmes cités en précision et en complexité de calcul, ils montrent aussi l'intérêt d'utiliser la technique SFS pour résoudre le problème de BM.

Chapitre IV : Conception et validation d'une nouvelle technique méta-heuristique : LBPO

Sommaire

IV.1 Introduction

Dans ce chapitre, nous développons un nouvel algorithme métaheuristique inspiré du concept de base du descripteur LBP (local binary pattern) et nommé optimisation à base de LBP (LBPO : local binary pattern optimizer). La méthode LBPO commence la recherche avec une population initiale des solutions candidates, ensuite et suivant le concept de base de LBP, un code spécifique est attribué à chaque solution. À partir de ces codes, l'algorithme LBPO sélectionne les meilleures solutions et met à jour la population. La nouveauté de cette technique est que le choix des meilleures solutions ne compte pas sur les valeurs de fitness mais sur le code donné à chaque solution, ce qui permet l'identification des meilleures solutions dans les différentes régions de l'espace de recherche, afin d'assurer un balancement effectif entre l'exploration et exploitation.

Dans la suite de ce chapitre, nous donnons une présentation détaillée sur notre méthode LBPO. Puis, nous évaluons l'algorithme avec 23 fonctions de test, nous comparons ensuite les résultats à ceux des autres métaheuristiques. À cet effet, dans le cadre de l'estimation de mouvement, un nouvel algorithme de mise en correspondance de blocs (BM) basé sur LBPO est proposé. Les performances de notre LBPO-BM en termes de précision et de complexité de calcul, sont évaluées avec des séquences vidéos très connues et comparées avec différentes techniques BM.

IV.2 Le descripteur LBP

Le descripteur LBP (local binary pattern) proposé par Ojala, et al. en 1994 [221, 222] est un opérateur très populaire en traitement d'image. Il consiste à générer pour chaque pixel dans l'image un code binaire, en comparant la valeur d'intensité du pixel central avec celles de ses voisins. Mathématiquement, l'indice LBP est calculé par l'équation.IV.1.

$$LBP = \sum_{q=0}^{Q-1} s(g_p - g_c)2^q \tag{IV.1}$$

Pour la version originale du LBP, qui utilise un voisinage de type 3×3, $Q = 8$. g_p est l'intensité du pixel voisin et g_c est celle du pixel central. Les étapes principales de LBP, illustrées dans la figure IV.1, sont :

1. Premièrement, l'algorithme soustrait l'intensité du pixel central de celles de ses voisins.

2. Les résultats sont représentés par des nombres binaires, où un résultat de soustraction négatif est codé par un "0" et les autres par un "1".

$$s(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases} \tag{IV.2}$$

3. Les nombres obtenus sont multipliés par les poids correspondants.

4. La somme de ces nouveaux nombres est calculée pour obtenir, dans cet exemple, le nombre 208 [221, 223].

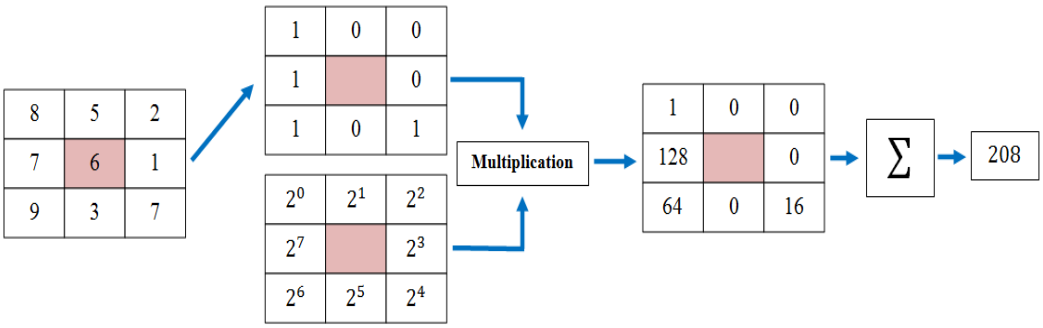


FIGURE IV.1. : Exemple de l'opérateur LBP.

De nombreuses variantes de LBP ont été proposés dans la littérature, pour améliorer ses performances et élargir le champ d'application comme [221], [222], [224]-[235].

Les propriétés les plus intéressantes de LBP sont :

- Une performance discriminante élevée ;
- L'invariance aux changements globaux d'illumination ;
- La simplicité de calcul.

La version originale de LBP a été initialement proposée pour caractériser la texture

d'une image, puis elle a été utilisée dans des applications diverses telles que l'analyse d'image, la reconnaissance d'objet, la biométrie, etc [236, 223, 237].

IV.3 La technique LBPO

L'efficacité du descripteur LBP s'explique par le fait que LBP ne dépend que des différences entre l'intensité d'un pixel et celles de ses voisins, ce qui permet de caractériser les détails. C'est ce qui nous a motivées à utiliser cet opérateur dans le contexte d'optimisation pour caractériser les différentes régions dans l'espace de recherche. Une nouvelle technique métaheuristique, nommée LBPO, inspirée par LBP est proposée dans ce chapitre. Nous expliquons par la suite la mise en œuvre de LBPO.
Les principales étapes de la technique LBPO sont les suivantes :

IV.3.1 Initialisation

La technique LBPO commence la recherche avec un ensemble de solutions candidates générées aléatoirement dans l'espace de recherche à l'aide de l'équation (IV.3).

$$X = (L_{max} - L_{min}) \times rand + L_{min} \tag{IV.3}$$

Où
L_{max} et L_{min} sont les bornes supérieure et inférieure de l'espace de recherche respectivement, et *rand* est un nombre aléatoire tiré uniformément dans l'intervalle [0, 1]. La matrice de toutes les solutions candidates peut être exprimée comme suit :

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,D} \\ x_{3,1} & x_{3,2} & \cdots & x_{3,j} & \cdots & x_{3,D} \\ \cdot & \cdot & \cdot & \cdots & \cdots & \cdot \\ x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,D} \\ \cdot & \cdot & \cdot & \cdots & \cdots & \cdot \\ x_{N_p,1} & x_{N_p,2} & \cdots & x_{N_p,j} & \cdots & x_{N_p,D} \end{bmatrix} \tag{IV.4}$$

D et N_p indiquent le nombre de dimensions et la taille de la population respectivement.

IV.3.2 Évaluation

Chaque solution candidate est évaluée par une fonction de fitness F, et les valeurs de fitness obtenues sont stockées dans un vecteur FX :

$$FX = \begin{bmatrix} F(x_1) \\ F(x_2) \\ . \\ . \\ F(x_i) \\ . \\ . \\ F(x_{N_p}) \end{bmatrix} \tag{IV.5}$$

IV.3.3 Détermination des solutions voisines

Pour chaque solution candidate, nous déterminons les huit solutions les plus proches en fonction de la distance euclidienne. Puis, on les classe dans l'ordre croissant selon la distance comme illustré dans la figure.IV.2.

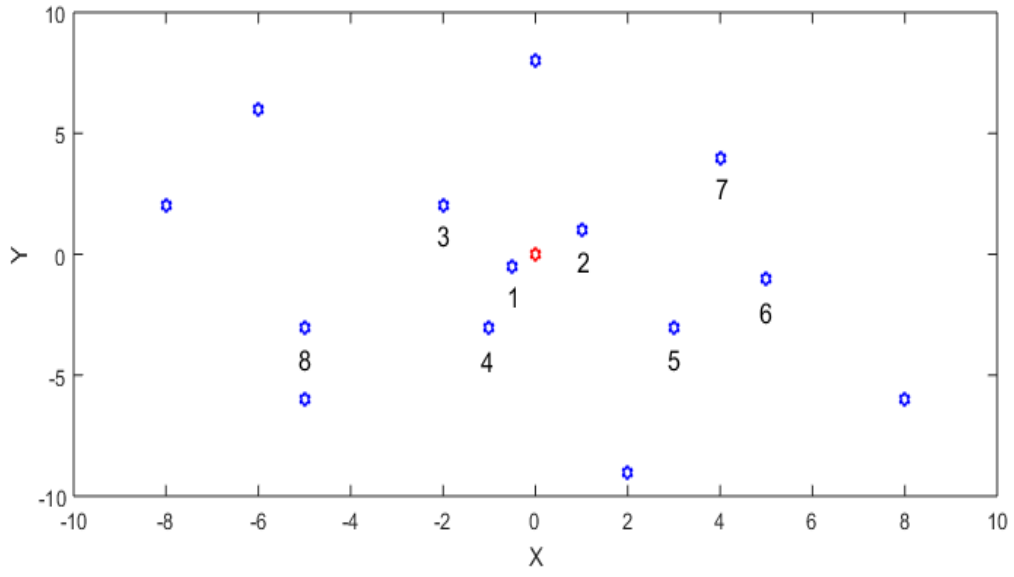


FIGURE IV.2. : Les solutions voisines d'une solution candidate : La solution candidate est indiquée en rouge et ses voisines en bleu.

Nous supposons également qu'il existe pour chaque i-ème solution candidate une matrice N_i contenant les solutions voisines et un vecteur FN_i comprenant les valeurs de fitness correspondantes.

$$N_i = \begin{bmatrix} n_{1,1} & n_{1,2} & \cdots & n_{1,j} & \cdots & n_{1,D} \\ n_{2,1} & n_{2,2} & \cdots & n_{2,j} & \cdots & n_{2,D} \\ n_{3,1} & n_{3,2} & \cdots & n_{3,j} & \cdots & n_{3,D} \\ . & . & . & \cdots & \cdots & . \\ . & . & . & \cdots & \cdots & . \\ n_{8,1} & n_{8,2} & \cdots & n_{8,j} & \cdots & n_{8,D} \end{bmatrix} \tag{IV.6}$$

$$FN_i = \begin{bmatrix} F(n_1) \\ F(n_2) \\ F(n_3) \\ . \\ . \\ F(n_8) \end{bmatrix} \tag{IV.7}$$

IV.3.4 Génération des codes

L'algorithme LBPO génère un code binaire $b_{i,j}$, $j = 1,2,...8$ pour chaque solution candidate, en comparant sa valeur de fitness avec celles de ses voisines.
Les nombres binaires générés sont multipliés par les poids correspondants, puis les résultats de multiplication sont additionnés pour obtenir un nombre entier d_i pour chaque i-ème solution candidate en utilisant l'équation.IV.8 et les deux équations IV.9 IV.10 selon le type du problème ; maximisation ou minimisation (voir figure IV.3).
Donc, l'algorithme LBPO donne des valeurs d_i trop petites pour les meilleures solutions, et des grandes valeurs pour les mauvaises solutions.

$$d_i = \sum_{q=0}^{Q-1} s(F(n_j) - F(x_i))2^q \tag{IV.8}$$

$$s(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases} : Maximisation \tag{IV.9}$$

$$s(z) = \begin{cases} 0, & z \geq 0 \\ 1, & z < 0 \end{cases} : Minimisation \tag{IV.10}$$

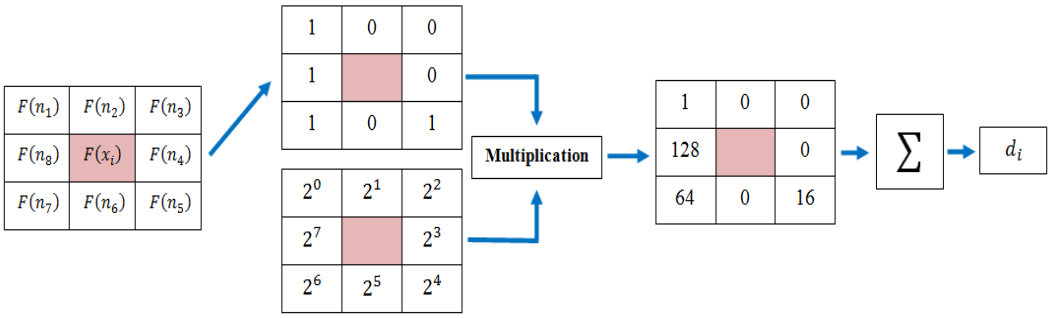


FIGURE IV.3. : Exemple de génération des codes d_i.

IV.3.5 Calcul des probabilités

Une valeur de probabilité Pr_i pour chaque i-ème solution candidate est calculée en fonction de son code d_i et en utilisant les équations IV.11 et IV.12

$$P_i = exp(-\beta_L \cdot d_i) \tag{IV.11}$$

et

$$Pr_i = \frac{P_i}{\sum_{i=1}^{N_p} P_i} \tag{IV.12}$$

où β_L est fixé, empiriquement, à 4.
Donc, l'algorithme LBPO donne des valeurs de probabilités Pr_i grandes pour les meilleures solutions, et des petites valeurs de Pr_i pour les mauvaises solutions.

IV.3.6 Mise à jour de la population

La population est mise à jour en utilisant les équations (IV.13) et (IV.14).

$$x_i^{iter+1} = x_r^{iter} - \omega_L \times \Delta x \tag{IV.13}$$

$$\Delta x = 2 \times rand \times (x_r^{iter} - x_i^{iter}) \tag{IV.14}$$

x_r^{iter} est une solution candidate sélectionnée par la méthode roulette [238]. La méthode de sélection roulette sélectionne les meilleures solutions, en utilisant les valeurs de probabilité Pr_i, (Elle est expliquée davantage dans l'annexe A) .

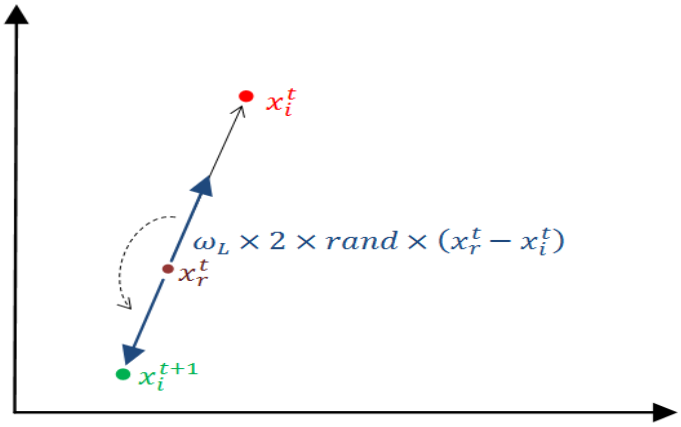


FIGURE IV.4. : Exemple de mise à jour de la position d'une solution

La figure. IV.4 illustre un exemple de mise à jour de la position d'une solution candidate. La solution x_i^{iter} se dirige progressivement vers la solution sélectionnée x_r^{iter}. Le paramètre ω_L est utilisé pour ajouter un aspect stochastique dans l'algorithme et balancer la recherche entre l'exploration et l'exploitation (voir figure IV.5). ω_L est donné par l'équation.IV.15 :

$$\omega_L = c \times a_L \times rand - a_L \tag{IV.15}$$

Le cœfficient a_L prend une valeur décroissante de 2 à 0 en utilisant l'équation suivante :

$$a_L = c - \frac{iter}{iter_{max}} \times c \tag{IV.16}$$

où $iter_{max}$ indique le nombre maximal d'itérations. Les processus de LBPO sont répétés jusqu'au nombre $iter_{max}$.

Le fait que les solutions candidates se déplacent vers des solutions **potentielles**, permet à l'algorithme LBPO d'exploiter les zones prometteuses de l'espace de recherche. Et comme les solutions se déplacent vers **différentes** solutions potentielles, et non seulement la meilleure solution globale, donc l'algorithme LBPO a la possibilité de visiter différentes régions et d'effectuer une exploration complète de l'espace de recherche.

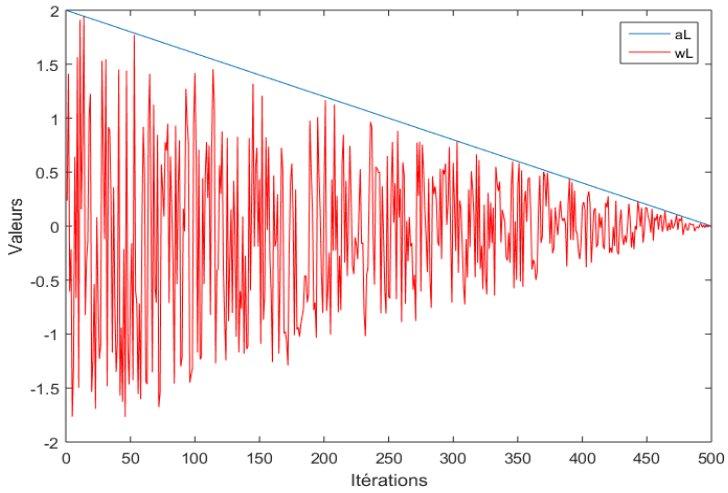


FIGURE IV.5. : Les variations de a_L et ω_L durant les itérations.

L'algorithme 7 donne un pseudo-code de la méthode LBPO.

Initialisation de la taille de population N_p,
Initialisation du nombre maximal d'itérations,
Initialisation aléatoire des solutions candidates,
Pour *iter* **de** 1 **à** le nombre d'itérations maximal **faire**
    Évaluation des solutions candidates par la fonction de fitness
    Générer les codes par l'équation IV.8
    Calculer les valeurs de probabilités par l'équation IV.12
    Mise à jour de la population en utilisant l'équation (IV.13)
Fin Pour

Algorithme 7 : Algorithme LBPO

IV.4 Analyse de LBPO

Dans cette section, nous étudions le comportement de la technique LBPO par l'analyse des trois courbes.

La première courbe représente la trajectoire d'une solution candidate (figure. IV.6), tracée par ses positions en fonction du nombre d'itérations. Nous pouvons constater qu'au départ, la solution prend des positions très éloignées afin d'explorer une vaste zone de l'espace de recherche, puis les modifications apportées à la position, sont réduit progressivement, pour fournir une recherche locale plus fine. Ce comportement passe l'algorithme LBPO du mode exploration au mode exploitation.

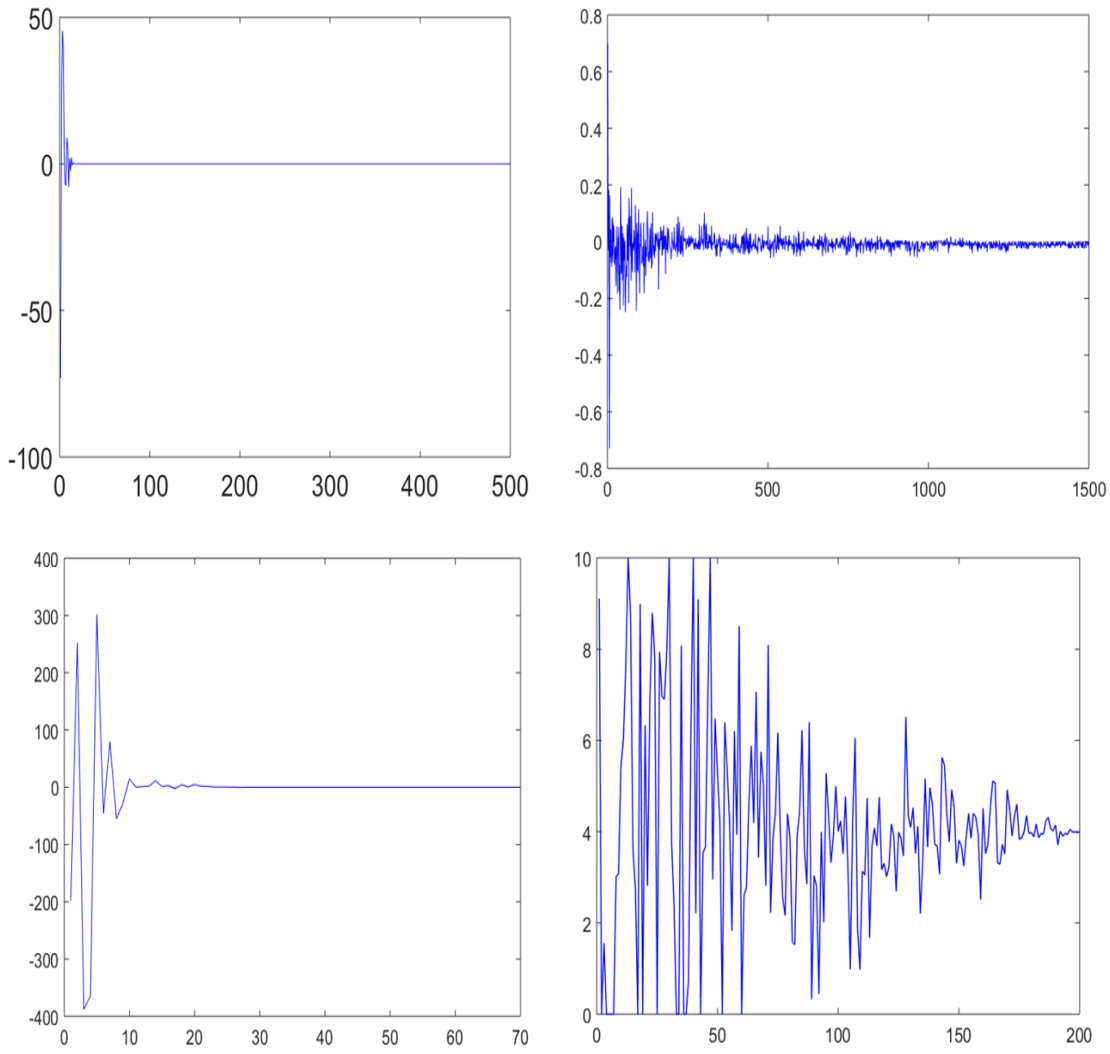


FIGURE IV.6. : La trajectoire d'une solution candidate obtenue avec les fonctions : F_{01}, F_{07}, F_{11} et F_{21}

La deuxième courbe représente la valeur de fitness moyenne des solutions candidate en fonction du nombre d'itérations (figure. IV.7). Il est clair que la valeur moyenne décroît de façon itérative au cours du processus d'optimisation, ce qui prouve que l'algorithme LBPO peut optimiser toutes les solutions.

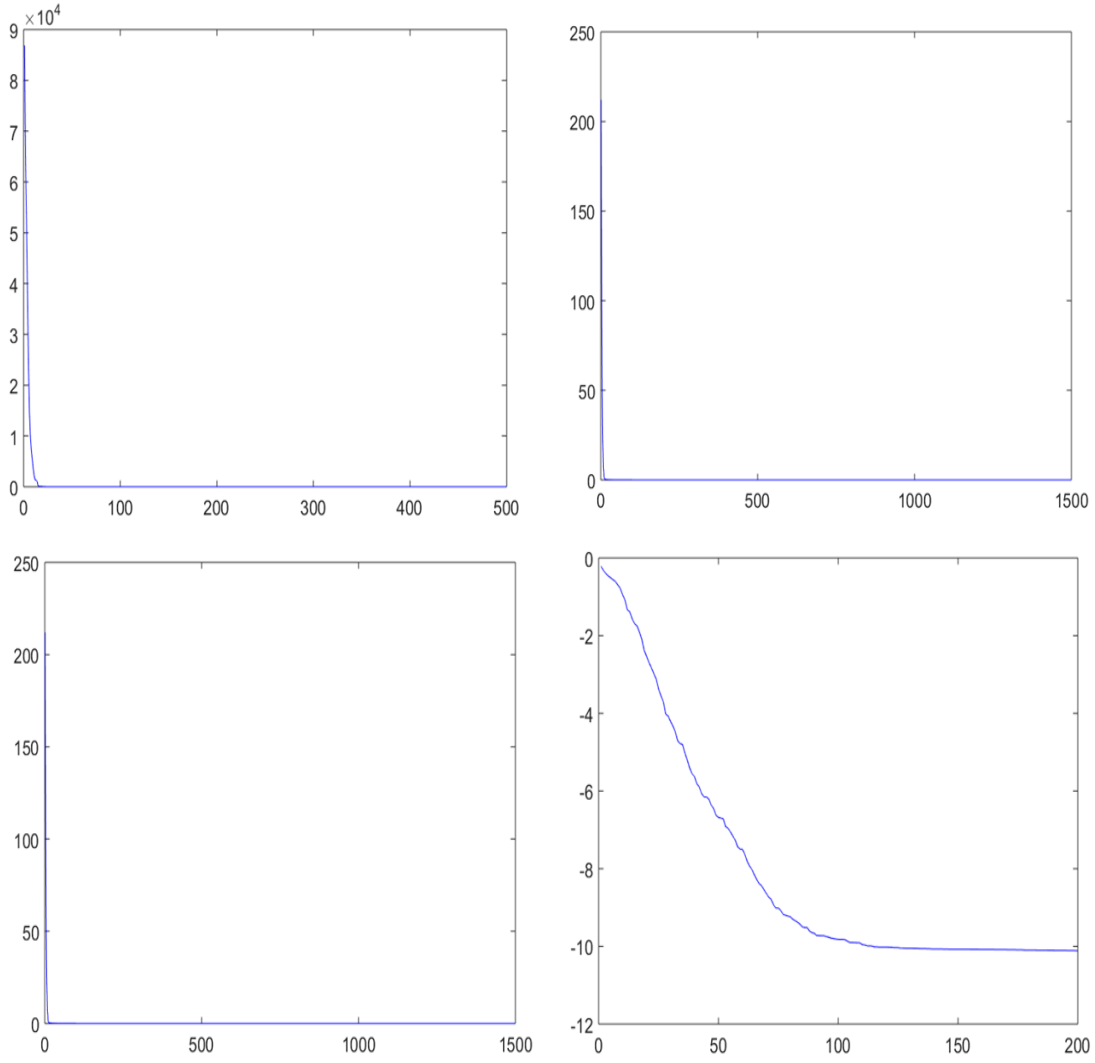


FIGURE IV.7. : Les courbes de la valeur de fitness moyenne obtenues avec les fonctions : F_{01}, F_{07}, F_{11} et F_{21}

La dernière est la courbe de convergence tracée par la meilleure valeur de fitness globale obtenue au cours des itérations (figure. IV.8). On peut remarquer que l'algorithme LBPO a un taux de convergence élevé, car il trouve rapidement la meilleure solution globale.

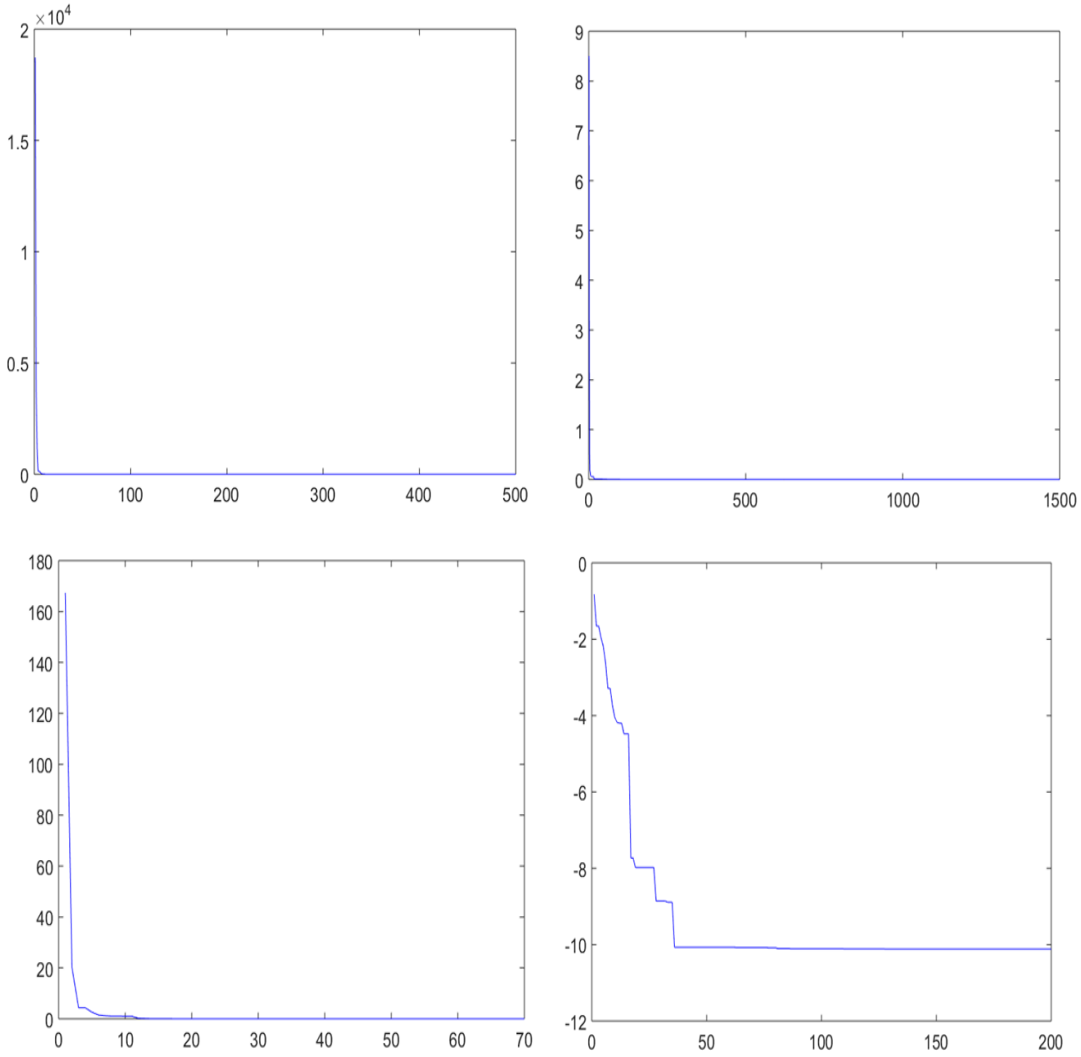


FIGURE IV.8. : Les courbes de convergence obtenues avec les fonctions : F_{01}, F_{07}, F_{11} et F_{21}

IV.5 Paramétrage

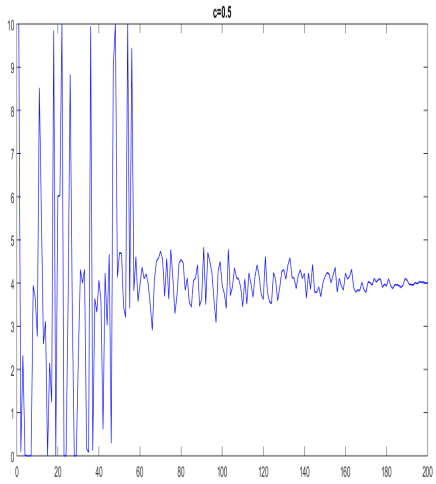
Le tableau IV.1 récapitule les valeurs des paramètres utilisés :

TABLE IV.1. : Les paramètres de l'algorithme LBPO

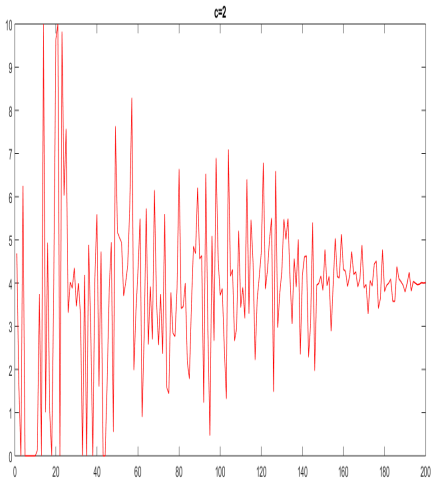
Paramètre	Description	Valeur
c	Utilisé dans les équations IV.15 et IV.16	2
$Iter_{max}$	Nombre d'itérations maximal	Selon la fonction de test (défini dans l'annexe B)
L_{max}	Borne maximal de l'espace de recherche	Selon la fonction de test
L_{min}	Borne minimal de l'espace de recherche	Selon la fonction de test
N_p	Taille de la population	100
N_{exc}	Nombre d'exécutions	25

IV.5.1 Influence du paramètre c

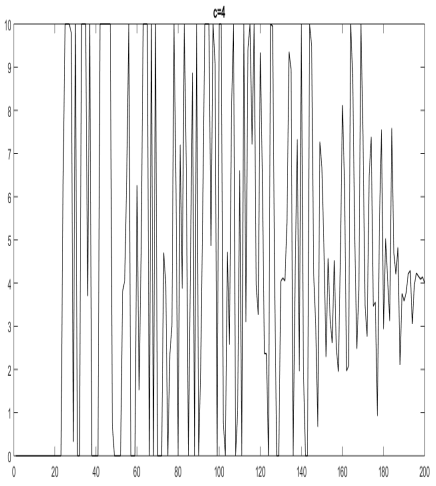
Pour étudier l'influence du paramètre c sur les performances de l'algorithme LBPO, ce dernier a été testé avec différentes valeurs de c : $c = 0.5$, $c = 2$, $c = 4$ et $c = 6$. Nous donnons dans la figure IV.9 les trajectoires d'une solution candidate pour les différentes valeurs de c. D'après les simulations réalisées, on voit clairement que pour une petite valeur de c ($c = 0.5$), la solution converge rapidement pour effectuer une recherche locale plus fine dans une petite région (exploitation). En incrémentant la valeur de c ($c = 2$), on peut balancer la recherche entre l'exploration et l'exploitation pour éviter la convergence prématurée. Par contre, trop augmenter la valeur de c ($c = 4$ et $c = 6$) peut nuire à l'exploitation de l'algorithme, et par conséquent l'algorithme diverge.



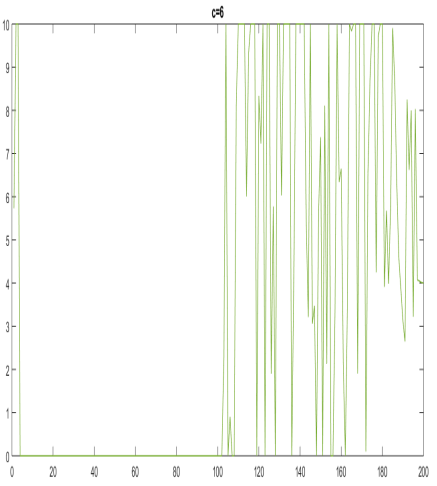
(a) $c = 0.5$



(b) $c = 2$



(c) $c = 4$



(d) $c = 6$

FIGURE IV.9. : La trajectoire d'une solution candidate obtenue avec différentes valeurs du paramètre c

IV.5.2 Influence de la taille de la population

Le choix de la taille de la population N_p est très important. Une taille de population trop petite peut nuire à la diversité de l'algorithme et conduire à l'obtention des résultats sous-optimal. Alors qu'une taille de population trop grande demandent un temps de calcul élevé et nécessite un espace mémoire considérable. Le tableau IV.2 présente les résultats obtenus par l'algorithme LBPO, pour trois fonctions de test différentes, en faisant varier la taille de la population. On remarque qu'une taille de population égale à 100 solutions candidates peut être la plus favorable. On constate également que l'augmentation de la taille de population ($N_p = 200$ et $N_p = 400$) ne donne pas une amélioration significative des performances, il est inutile d'augmenter la valeur N_p plus que 100.

TABLE IV.2. : Influence de la taille de la population N_p

N_p	F_{01}	F_{09}	F_{15}
20	$4,31.10^{-188}$	$8,05.10^{-8}$	0,0063
50	$8,18.10^{-204}$	$5,94.10^{-7}$	0,001
100	$4,61.10^{-207}$	$1,44.10^{-4}$	$7,02.10^{-4}$
200	$9,84.10^{-194}$	0,017	$4,61.10^{-4}$
400	$1,96.10^{-192}$	0,183	$3,17.10^{-4}$

IV.5.3 Complexité de calcul

Le temps de calcul d'un algorithme est liée à la structure de l'algorithme, et elle dépend aussi de la puissance de l'ordinateur. Couramment, nous utilisons la notations grande O, pour exprimer la complexité de calcul. La complexité de calcul de la technique LBPO réside essentiellement dans le processus de mise à jour, la génération des codes et dans le calcul des probabilités. Elle dépend aussi au nombre de solutions candidates N_p, aux dimensions D et au nombre maximal d'itérations $iter_{max}$. Elle peut être calculée comme suit :

$$\begin{aligned} O(LBPO)= \quad & O(Processus\ de\ mise\ a\ jour)+O(Generation\ de\ code) \\ & +O(Calcul\ de\ probabilites) \end{aligned} \tag{IV.17}$$

$$O(LBPO) = O(N_p \times D \times iter_{max}) + O(N_p \times iter_{max}) + O(N_p \times iter_{max}) \tag{IV.18}$$

IV.6 Évaluation de LBPO

Nous présentons dans cette section les résultats d'optimisation obtenus par notre méthode LBPO en utilisant les 23 fonctions de test (*CEC'05*) présentées dans la section II.3.5.1. Les performances de LBPO sont comparées à celles d'autres métaheuristiques. L'intérêt de cette évaluation est de prouver que la méthode LBPO donne des résultats satisfaisants dépassent d'autres techniques, dans des cas différents.

La méthode LBPO a été mise en œuvre en utilisant Matlab 2015. Deux valeurs sont étudiées : la valeur moyenne (μ) obtenue sur un nombre fixe d'exécutions ($N_{exc} = 25$) et l'écart type (σ) entre les différentes exécutions. Les résultats sont comparés avec sept méthodes métaheuristiques : CA [240] (1994), PSO [172] (1995), BH [177] (2013), GWO [176] (2014), SCA [161] (2016) et SSA [241] (2017), où les codes de ces techniques ont été tirés des liens mentionnés dans l'annexe D.
Deux types de tests sont définis :

1. Dans le premier test, les expériences sont effectuées avec les 23 fonctions de test.

2. Dans le deuxième test, nous évaluons l'effet sur les performances de LBPO si nous augmentons les dimensions des fonctions.

IV.6.1 Expérimentation 1

Dans cette section, la technique LBPO est testée avec 23 fonctions de test, les résultats de simulation sont présentés dans les tableaux.IV.3, IV.4 et IV.5 correspondent aux valeurs moyennes (μ), l'écart type (σ). Les algorithmes sont également classés (Rank) selon les performances pour chaque fonction, et le classement moyen et global sont ensuite déterminés.

D'après les résultats, l'algorithme proposé LBPO dépasse toutes les autres techniques pour la plupart des fonctions de test :

- Pour les fonctions uni-modales, notre algorithme LBPO dépasse les autres métaheuristiques pour toutes les fonctions F_{01}, F_{02}, F_{03}, F_{04}, F_{06}, F_{07} excepté la fonction F_{05}.

La fonction F_{05} appelée la fonction *Rosenbrock* est une fonction uni-modale, elle a été

proposée par Rosenbrock en 1960. Cette fonction a un optimum global à l'intérieur d'un longue vallée étroite de forme parabolique [171]. Trouver la vallée est facile, mais, converger vers le minimum global est difficile. Toutes les techniques d'optimisation trouvent des difficultés à résoudre ce type de problème.

- Selon les résultats obtenus avec les fonctions multimodales, l'algorithme LBPO est supérieur aux autres algorithmes dans les fonctions F_{09}, F_{10}, F_{11}, F_{15} et F_{21}. Il atteint la valeur optimale théorique avec la fonction F_{11}.

  La fonction F_{11} est une fonction hautement multimodale, car le nombre des minima croît de façon exponentielle quand le nombre de dimensions augmente. La résolution parfaite (optimum global atteint) de cette fonction en utilisant la technique LBPO, valide davantage la supériorité de notre technique.

- LBPO donne également des résultats acceptables avec les fonctions F_{14}, F_{22} et F_{23} et il est classé en deuxième place pour ces fonctions.

Les résultats avec les fonctions de test montrent et prouvent les capacités d'exploitation et d'exploration de l'algorithme LBPO proposé.

La figure IV.10 illustre les courbes de convergence obtenues avec les fonctions F_{01}, F_{04}, F_{09}, F_{10}, F_{11} et F_{19}. Il est clair que l'algorithme LBPO converge plus rapidement que les autres méthodes.

La figure IV.11 illustre les diagrammes en boîtes pour les fonctions F_{01}, F_{04}, F_{09}, F_{10}, F_{11} et F_{19}, où la ligne rouge représente la valeur médiane des 25 exécutions indépendantes. Les côtés de la boîte bas et haut représentent respectivement les valeurs de fitness minimale et maximale. L'écart type est la distance entre ces cotés. Nous pouvons constater que l'algorithme LBPO donne un écart type plus petit que d'autres méthodes, ce qui confirme sa stabilité et sa robustesse.

TABLE IV.3. : Résultats de LBPO et d'autres métaheuristiques pour les fonctions unimodales à haute dimension

Fonction		**Optimum**	**CA**	**PSO**	**BH**	**GWO**	**SCA**	**SSA**	**LBPO**
F_{01}	μ	0	$1,29.10^{-4}$	0,39	0,57	$8,13.10^{-41}$	1,27	$1,10.10^{-8}$	$4,61.10^{-207}$
	σ		$1,3.10^{-4}$	0,32	0,55	$1,17.10^{-40}$	3,32	$2,47.10^{-9}$	0
	Rank		4	5	6	2	7	3	1
F_{02}	μ	0	59,22	0,03	2,44	$8,03.10^{-47}$	$2,39.10^{-6}$	0,3390	$4,00.10^{-196}$
	σ		20.97	0,16	1,13	$8,34.10^{-47}$	$6,62.10^{-6}$	0,4880	0
	Rank		7	4	6	2	3	5	1
F_{03}	μ	0	$2,64.10^{4}$	$1,64.10^{3}$	$2,87.10^{2}$	$2,86.10^{-11}$	$3,80.10^{3}$	85,042	$4,39.10^{-197}$
	σ		$4,62.10^{3}$	$6,40.10^{2}$	61.76	$7,87.10^{-11}$	$3,07.10^{3}$	62,983	0
	Rank		7	6	4	2	5	3	1
F_{04}	μ	0	3,13	2,19	9,24	$1,38.10^{-21}$	5,38	0,9168	$5,45.10^{-205}$
	σ		3,24	1,71	1,94	$2,85.10^{-21}$	5,90	0,6857	0
	Rank		5	4	7	2	6	3	1
F_{05}	μ	0	48,56	33,01	35,29	25,898	27,05	46,695	28,82
	σ		$1,15.10^{2}$	31,93	30,37	0,7736	0,76	44,217	0,02
	Rank		7	4	5	1	2	6	3
F_{06}	μ	0	$1,56.10^{4}$	$2,76.10^{4}$	$4,37.10^{3}$	$1,23.10^{+3}$	$3,11.10^{4}$	3240,37	6,61
	σ		$3,30.10^{3}$	$4,94.10^{3}$	$1,25.10^{3}$	330,21	$5,91.10^{3}$	856,889	2,11
	Rank		5	6	4	2	7	3	1
F_{07}	μ	0	0,07	0,02	0.03	$1,94.10^{-4}$	$8,6.10^{-3}$	0,0209	$4,90.10^{-5}$
	σ		$1,77.10^{-2}$	0,01	$1,09.10^{-2}$	$9,66.10^{-5}$	$8,5.10^{-3}$	0,0074	$2,50.10^{-5}$
	Rank		7	5	6	2	3	4	1

TABLE IV.4. : Résultats de LBPO et d'autres métaheuristiques pour les fonctions multimodales à haute dimension

Fonction		**Optimum**	**CA**	**PSO**	**BH**	**GWO**	**SCA**	**SSA**	**LBPO**
F_{08}	μ	-418,9829 * n	$-1,04.10^{4}$	$-9,94.10^{3}$	$-4,51.10^{3}$	$-6,34.10^{+3}$	$-4,33.10^{3}$	$-7,5458.10^{3}$	$-6,98.10^{3}$
	σ		$6,69.10^{2}$	$5,14.10^{2}$	$4,48.10^{2}$	818,94	$2,85.10^{2}$	766,15	$1,41.10^{3}$
	Rank		1	2	6	5	7	3	4
F_{09}	μ	0	$3,16.10^{2}$	$2,75.10^{2}$	$1,95.10^{2}$	47,608	$2,31.10^{2}$	110,16	$1,44.10^{-4}$
	σ		18,63	17,24	21,72	15,101	47,08	12,558	$6,15.10^{-4}$
	Rank		7	6	4	2	5	3	1
F_{10}	μ	$8,8818.10^{-16}$	11,18	13,44	8,24	0,0844	16,79	6,6451	$6,48.10^{-8}$
	σ		0,88	0,73	0,90	0,0292	3,87	0,9805	$1,62.10^{-7}$
	Rank		5	6	4	2	7	3	1
F_{11}	μ	0	18,18	43,50	8,01	0,084	36,24	4,29449	0
	σ		4,63	11,94	2,14	0,0495	15,66	1,1682	0
	Rank		5	7	4	2	6	3	1
F_{12}	μ	$1,5705.10^{-32}$	4,19	0,04	5,34	0,0120	0,36	2,2093	0,24
	σ		5,98	0,09	2,71	0,0119	0,05	1,3512	0,08
	Rank		6	2	7	1	4	5	3
F_{13}	μ	0	$6,2.10^{-3}$	$4,0.10^{-3}$	1,32	0,2046	2,03	0,0021	2,44
	σ		$9,6.10^{-3}$	$9,5.10^{-3}$	2,33	0,1377	0,16	0,0044	0,43
	Rank		3	2	5	4	6	1	7

TABLE IV.5. : Résultats de LBPO et d'autres métaheuristiques pour les fonctions multimodales à faible dimension

Fonction		**Optimum**	**CA**	**PSO**	**BH**	**GWO**	**SCA**	**SSA**	**LBPO**
F_{14}	μ	0,998004	1,15	0,9980	1,27	3,3878	1,79	1,3553	1,07
	σ		0,79	$4,35.10^{-13}$	0,60	0,6946	0,99	0,6946	0,39
	Rank		3	1	4	7	6	5	2
F_{15}	μ	$3,075.10^{-4}$	$44,0.10^{-4}$	$36,0.10^{-4}$	$7,88.10^{-4}$	0,0019	$9,56.10^{-4}$	0,0008	$7,02.10^{-4}$
	σ		$12,2.10^{-3}$	$75,0.10^{-4}$	$3,94.10^{-4}$	0,0056	$3,94.10^{-4}$	0,0003	$4,04.10^{-4}$
	Rank		7	6	2	5	4	3	1
F_{16}	μ	-1,0316285	-1,0316	-1,0316	-1,0316	-1,0316	-1,0316	-1,0316	-1,0316
	σ		$6,24.10^{-16}$	$7,06.10^{-13}$	$4,70.10^{-12}$	$1,69.10^{-8}$	$5,28.10^{-5}$	$3,48.10^{-14}$	$7,33.10^{-6}$
	Rank		1	3	4	5	7	2	6
F_{17}	μ	0,398	0,3979	0,3979	0,3979	0,3979	0,3999	0,397887	0,397888
	σ		0	$1,01.10^{-11}$	$3,81.10^{-9}$	$1,59.10^{-6}$	$17,0.10^{-4}$	$3,15.10^{-14}$	$1,78.10^{-6}$
	Rank		3	1	23	4	7	5	6
F_{18}	μ	3	3,0000	3,0000	3,0000	3,0000	3,0000	3	3,0000
	σ		$1,36.10^{-15}$	$1,89.10^{-12}$	$5,93.10^{-11}$	$2,11.10^{-5}$	$9,33.10^{-5}$	$2,60.10^{-13}$	$3,01.10^{-5}$
	Rank		1	3	4	5	7	2	6
F_{19}	μ	-3,86	-3,8628	-3,8628	-3,8604	-3,8612	-3,8537	-3,86275	-3,8624
	σ		$1,84.10^{-15}$	$1,05.10^{-7}$	$24,0.10^{-4}$	0,0025	$37,0.10^{-4}$	$2,02.10^{-5}$	$8,07.10^{-4}$
	Rank		5	6	1	2	7	4	3
F_{20}	μ	-3,32	-3,2792	-3,2602	-3,2339	-3,2608	-3,0135	-3,2195	-3,2328
	σ		$58,2.10^{-3}$	$60,6.10^{-3}$	$86,1.10^{-3}$	0,0732	$205,7.10^{-3}$	0,0527	0,0485
	Rank		1	2	4	3	7	6	5
F_{21}	μ	-10,1532	-7,7627	-6,9258	-8,5495	-9,9496	-3,2472	-9,2546	-10,08
	σ		3,5566	3,0358	2,9690	1,0105	1,6226	2,4834	0,1029
	Rank		5	6	4	2	7	3	1
F_{22}	μ	-10,4029	-7,5721	-7,8961	-9,1585	-10,4009	-4,5168	-9,7684	-10,3429
	σ		3,5908	3,4918	2,5949	0,0011	1,4180	1,7537	0,0837
	Rank		6	5	4	1	7	3	2
F_{23}	μ	-10,5364	-7,7253	-8,5370	-9,6884	-10,3185	-4,4472	-8,3600	-10,1838
	σ		3,6025	3,3054	2,3828	1,0813	0,9653	3,3014	1,0774
	Rank		6	4	3	1	7	5	2
Rank moyen			4.56	4.21	4.39	2.78	5.82	3,60	**2,60**
Rank global			6	4	5	2	7	3	**1**

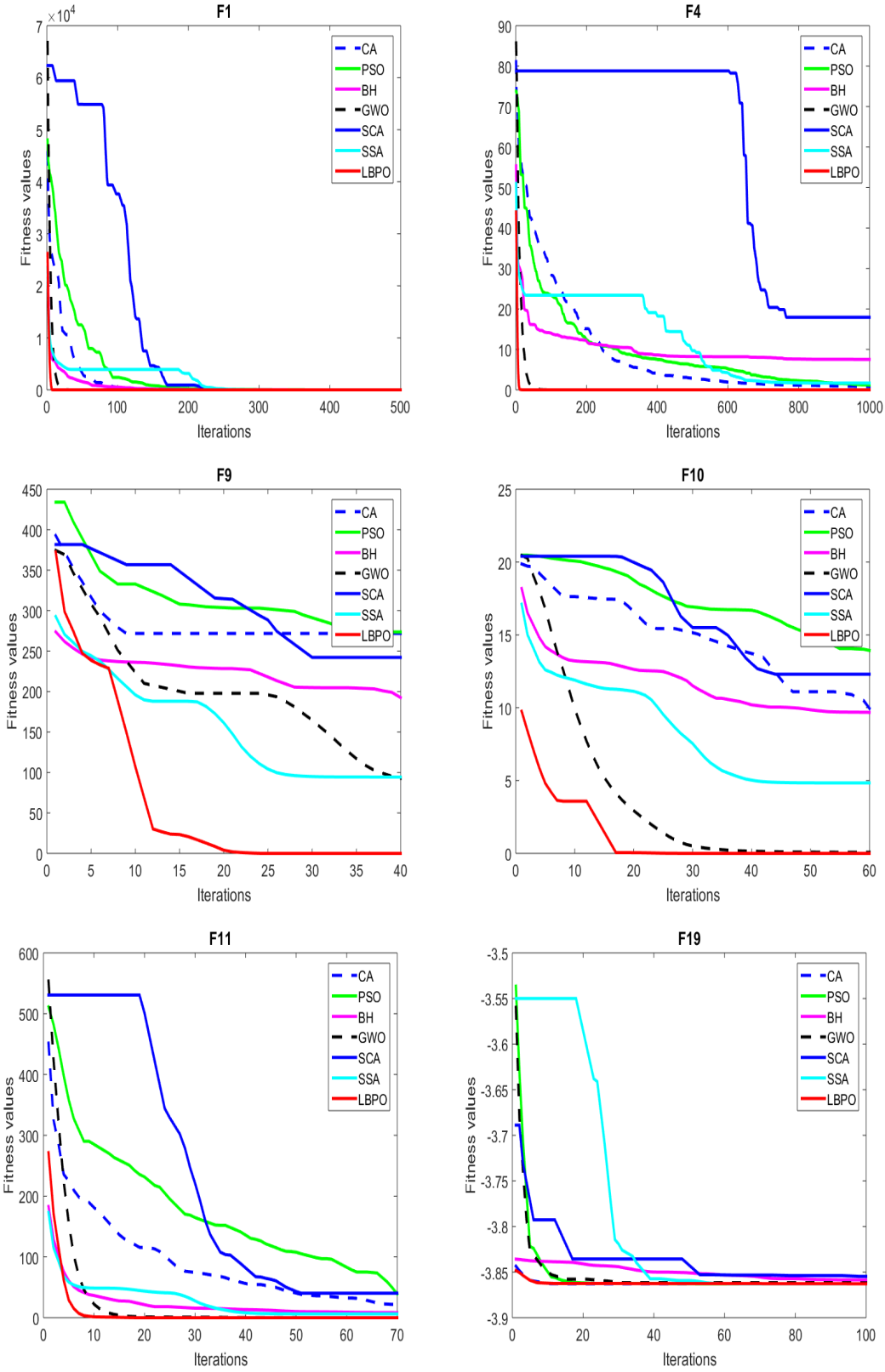


FIGURE IV.10. : Courbes de convergence obtenues avec les fonctions F_{01}, F_{04}, F_{09}, F_{10}, F_{11} et F_{19}

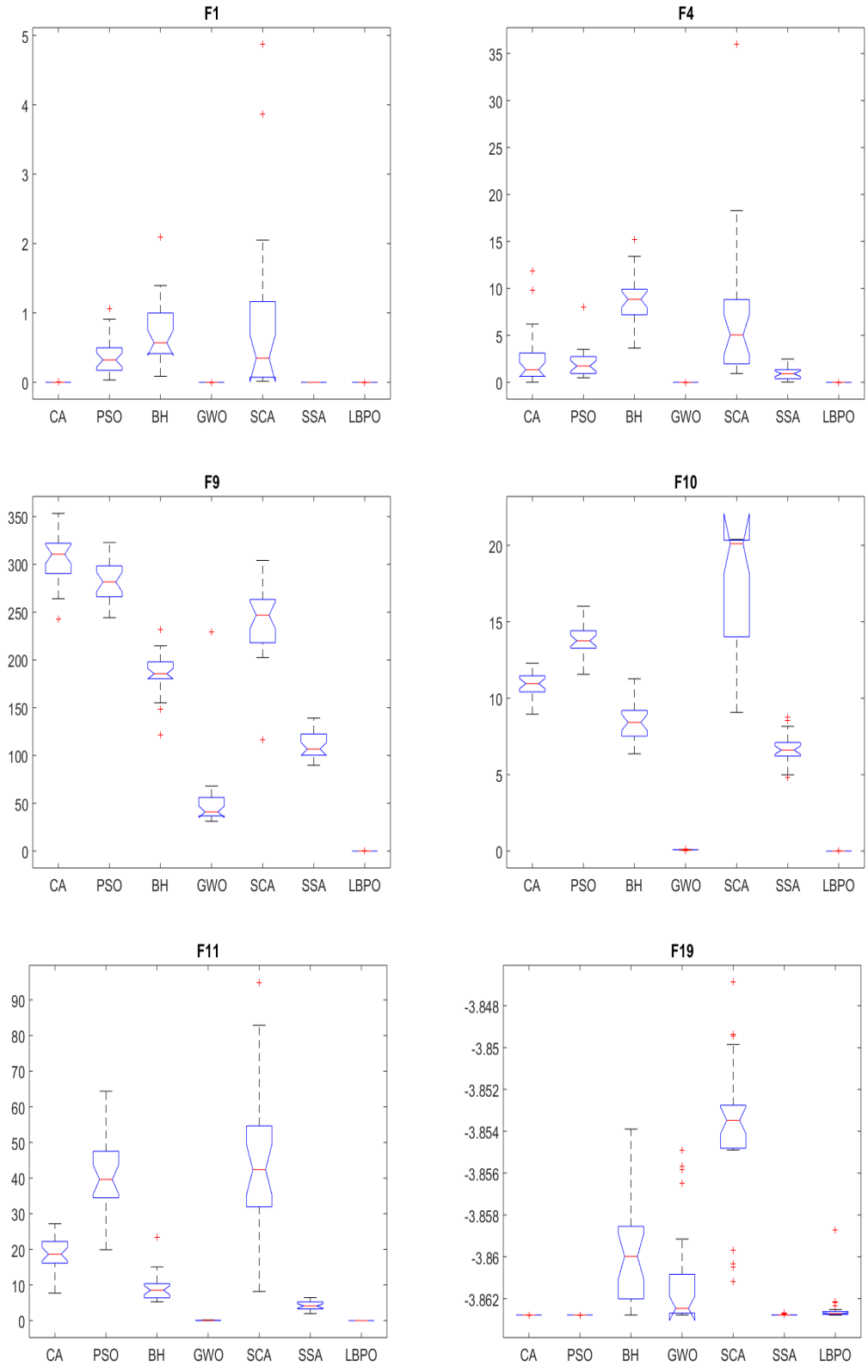


FIGURE IV.11. : Les diagramme en boite obtenues avec les fonctions F_{01}, F_{04}, F_{09}, F_{10}, F_{11} et F_{19}

IV.6.2 Expérimentation 2

Au cours des dernières années, la résolution des problèmes d'optimisation de très haute dimension à l'aide des techniques métaheuristiques, a suscité un intérêt croissant [242]. Il est connu que l'espace de recherche d'un problème augmente de façon exponentielle avec la taille du problème, ce qui entraîne une dégradation des performances de nombreuses métaheuristiques.

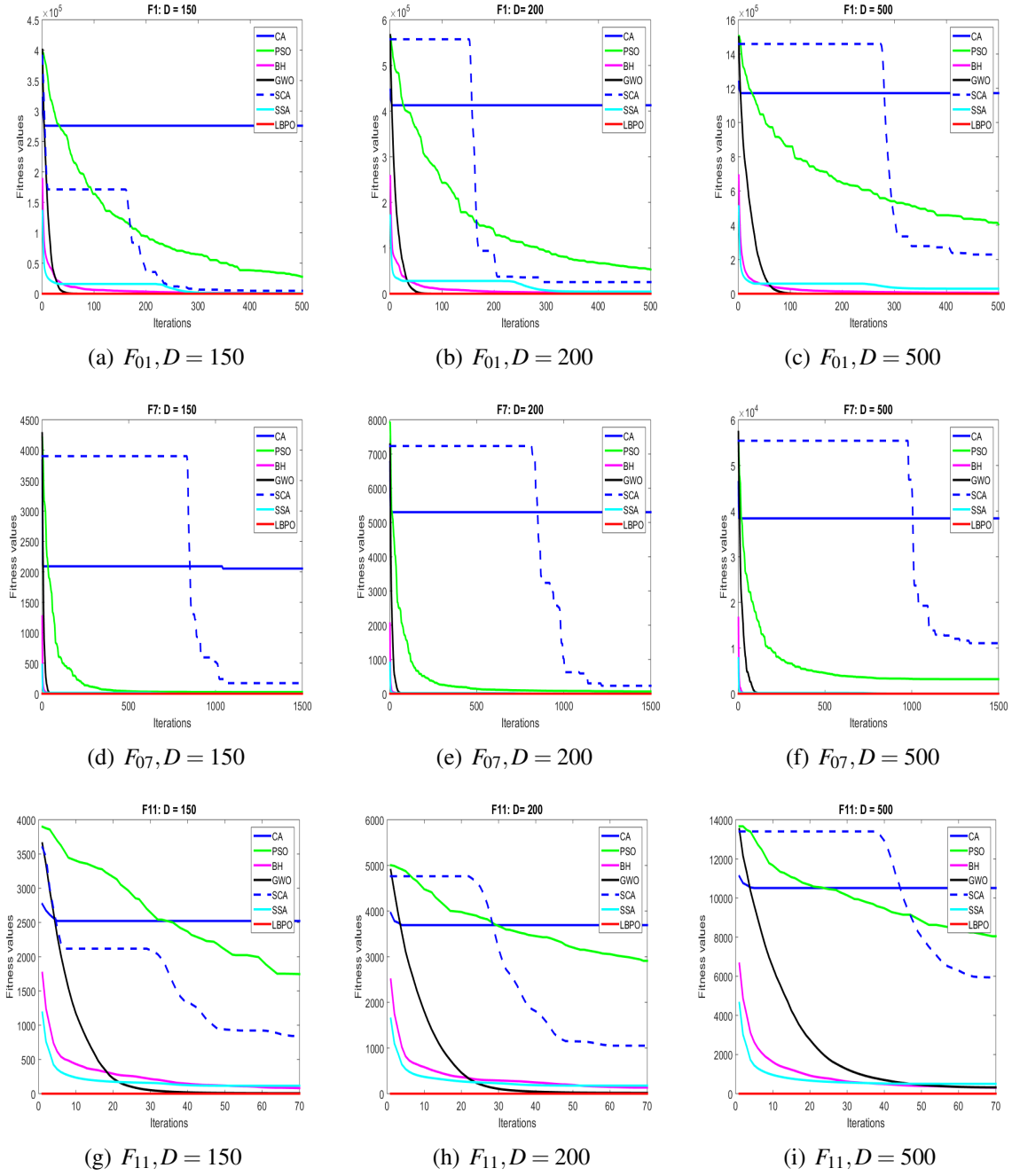
Dans cette expérimentation, nous étudions les performances de la technique LBPO sur des fonctions de test de très hautes dimensions. Les fonctions utilisées sont F_{01}, F_{04}, F_{07}, F_{10}, F_{11} et F_{12} avec les dimensions $D = 150$, $D = 200$ et $D = 500$. Les résultats obtenus sont comparés avec les autres métaheuristiques en terme de la valeur de fitness moyenne dans le tableau IV.6.

En comparant ces résultats, nous observons que chaque fois que la dimension augmente, les résultats de tous les algorithmes se détériorent. Cependant, contrairement aux autres métaheuristiques, la détérioration des performances de LBPO est insignifiante. Par exemple, avec F_{10}, les résultats sont $7.8590.10^{-9}$, $9.1857.10^{-8}$, et $1.6808.10^{-7}$ pour les dimensions 150, 200 et 500 respectivement. Il est évident que la dégradation des résultats de LBPO est très faible. On peut aussi remarquer qu'avec la fonction F_{11}, l'algorithme LBPO a obtenu la valeur optimale zéro quelle que soit la dimension du problème. De plus, l'algorithme LBPO maintient une performance supérieure à toutes les métaheuristiques citées, ce qui prouve encore l'efficacité de l'algorithme LBPO dans la résolution des problèmes de très hautes dimensions.

La figure IV.12 illustre les courbes de convergence de l'algorithme LBPO et les autres métaheuristiques obtenues avec les fonctions F_{01}, F_{07} et F_{11}, où les dimensions sont 150, 200 et 500. Il est évident que l'algorithme LBPO a une convergence plus rapide que tous les autres algorithmes.

TABLE IV.6. : Résultats de LBPO et d'autres métaheuristiques pour certaines fonctions unimodales et multimodales avec différentes dimensions

Fonctions	Dimensions	CA	PSO	BH	GWO	SCA	SSA	LBPO
F_1	D=30	$1,29.10^{-4}$	0,39	0,57	$8,13.10^{-41}$	1,27	$1,10.10^{-8}$	$4,61.10^{-207}$
	D=150	$2,7591.10^{+5}$	$2,7090.10^{+4}$	585.6315	$1,35.10^{-14}$	$2,3655.10^{+4}$	879.3569	$2.2817.10^{-204}$
	D=200	$3,9913.10^{+5}$	$5,8479.10^{+4}$	$1,0287.10^{+3}$	$1,69.10^{-11}$	$4,5585.10^{+4}$	$3,2135.10^{+3}$	$3,2825.10^{-207}$
	D=500	$1,1511.10^{+6}$	$3,6582.10^{+5}$	$4.4308.10^{+3}$	$1,15.10^{-5}$	$2,1786.10^{+5}$	$3,1407.10^{+4}$	$3,1724.10^{-208}$
F_4	D=30	3,13	2,19	9,24	$1,38.10^{-21}$	5,38	0,9168	$5,45.10^{-205}$
	D=150	96.5282	44.7789	18.4520	0,0021	91.1091	17.5658	$3,9721.10^{-207}$
	D=200	97.5589	50.0403	19.5595	1,7195	94.5629	19.4831	$5,3865.10^{-206}$
	D=500	98.9096	65.8538	21.8101	43,2951	98.4312	24.2307	$1,2160.10^{-207}$
F_7	D=30	0,07	0,02	0.03	$1,94.10^{-4}$	$8,6.10^{-3}$	0,0209	$4,90.10^{-5}$
	D=150	$2,3331.10^{+3}$	30.3293	0.3600	0,0002	187.7614	0.5817	$4,6049.10^{-5}$
	D=200	$5,0140.10^{+3}$	104.3172	0.4819	0,0004	571.1087	1.1808	$4,5485.10^{-5}$
	D=500	$4,1684.10^{+4}$	$3,2676.10^{+3}$	1.7395	0,0014	$9,2199.10^{+3}$	10.3994	$4,0166.10^{-5}$
F_{10}	D=30	11,18	13,44	8,24	0,0844	16,79	6,6451	$6,48.10^{-8}$
	D=150	20.4167	20.1489	10.6585	5,7004	20.4688	10.9255	$7,8590.10^{-9}$
	D=200	20.5241	20.3580	10.7658	6,4113	20.1123	11.2295	$9,1857.10^{-8}$
	D=500	20.7238	20.1711	10.9063	11,6815	20.4743	11.4743	$1,6808.10^{-7}$
F_{11}	D=30	18,18	43,50	8,01	0,084	36,24	4,2944	0
	D=150	$2,5247.10^{+3}$	$1,7103.10^{+3}$	89.2952	6,2109	$1,3745.10^{+3}$	123.6544	0
	D=200	$3,6018.10^{+3}$	$2,5323.10^{+3}$	127.83	16,0304	$1,9822.10^{+3}$	173.4953	0
	D=500	$1,0348.10^{+4}$	$8,2463.10^{+3}$	349.3191	319,37	$5,7833.10^{+3}$	487.6952	0
F_{12}	D=30	4,19	0,04	5,34	0,0120	0,36	2,2093	0,24
	D=150	$2,5680.10^{+9}$	$1,0774.10^{+6}$	3.2639	0,1428	$2,2997.10^{+8}$	7.2710	0.3020
	D=200	$3,9964.10^{+9}$	$1,6423.10^{+7}$	2.7384	0,3747	$5,3172.10^{+8}$	8.0110	0.2810
	D=500	$1,2878.10^{+10}$	$4,5022.10^{+8}$	2.0979	0,6609	$3,2423.10^{+9}$	14.7652	0.2946



(a) $F_{01}, D = 150$ (b) $F_{01}, D = 200$ (c) $F_{01}, D = 500$

(d) $F_{07}, D = 150$ (e) $F_{07}, D = 200$ (f) $F_{07}, D = 500$

(g) $F_{11}, D = 150$ (h) $F_{11}, D = 200$ (i) $F_{11}, D = 500$

FIGURE IV.12. : Courbes de convergence sur les dimensions 150, 200 et 500 obtenues avec F_{01}, F_{07} et F_{11}

IV.6.3 Analyse statistique

Pour vérifier si les différences entre les solutions trouvées par la technique LBPO et d'autres métaheuristiques étaient statistiquement significatives, nous avons effectué un test de Student bilatéral.
Le test de Student bilatéral est un test statistique qui permet de trancher entre deux hypothèses au vu des résultats des échantillons. Soient H_0 et H_1 deux hypothèses (H_0 est appelée hypothèse nulle, H_1 hypothèse alternative), une seule qui est vraie et l'autre est rejetée. Pour un test de Student bilatéral, nous émettons les hypothèses suivantes :

- Hypothèse nulle, $H_0 : m_1 = m_2$
- Hypothèse alternative, $H_1 : m_1 \neq m_2$.

Mathématiquement, on calcule la valeur statistique t en utilisant l'équation suivante :

$$t = \frac{\mu_1 - \mu_2}{\sqrt{(\frac{(\sigma_1)^2}{N_{exe}}) + (\frac{(\sigma_2)^2}{N_{exe}})}} \tag{IV.19}$$

Où μ_1 et σ_1 représentent la moyenne et l'écart type obtenus par l'algorithme 1 et μ_2 et σ_2 sont la moyenne et l'écart type obtenus par l'algorithme 2. N_{exe} est le nombre d'essais ou d'exécutions égal à 25.

Pour pouvoir rejeter ou accepter l'hypothèse nulle d'égalité des moyennes, la valeur statistique de t est comparée avec une valeur critique obtenue à partir de la table de la loi de Student, en fonction du seuil de probabilité α_p et du nombre de degrés de liberté $df = (N_{exe} + N_{exe} - 2)$ (un extrait de cette table est donné dans l'annexe C). Si la valeur absolue de la statistique t est égale ou supérieure à la valeur critique, l'hypothèse nulle est rejetée et l'hypothèse alternative est acceptée, c'est-à-dire il y a une différence significative entre les algorithmes [243, 171].
Avec le nombre d'exécutions $N_{exe} = 25$, le degré de liberté est $df = 25 + 25 - 2 = 48$, et pour un degré de confiance de 95 %, le seuil de probabilité $\alpha_p = 0.05$. En fonction de ces valeurs et à partir de la table de student la valeur critique est égale à 1.677.

Dans le tableau IV.7, nous présentons les résultats de la comparaison statistique entre l'algorithme LBPO et les autres techniques métaheuristiques ; la marque $\star$ signifie que la dif-

férence entre les deux techniques est statistiquement significative. D'après ces résultats, nous pouvons constater que la valeur statistique t dépasse la valeur critique pour les cas suivants :

- LBPO vs CA : 20 fonctions sur 23.
- LBPO vs PSO : 20 fonctions sur 23.
- LBPO vs BH : 18 fonctions sur 23.
- LBPO vs GWO : 20 fonctions sur 23.
- LBPO vs SCA : 22 fonctions sur 23.
- LBPO vs SSA : 18 fonctions sur 23.

Ce qui signifie que pour ces fonctions, l'hypothèse nulle est rejetée avec un degré de confiance de 95%. Ce qui prouve aussi que la différence entre les moyennes comparées est significative, et elle ne peut être due simplement au hasard.

TABLE IV.7. : Résultats de test de Student bilatéral

Fonction	**LBPO vs CA**	**LBPO vs PSO**	**LBPO vs BH**	**LBPO vs GWO**	**LBPO vs SCA**	**LBPO vs SSA**
F_1	-4,848⋆	-6,185 ⋆	-5,204⋆	-3,458 ⋆	-1,911⋆	-22,26⋆
F_2	-14,11⋆	-1,083	-10,75⋆	$-4,817$⋆	-1,809⋆	-3,473⋆
F_3	-28,59⋆	-12,83⋆	-23,28⋆	$-1,820$⋆	-6,181⋆	-6,751⋆
F_4	-4,830⋆	-6,423⋆	-23,75⋆	$-2,420$⋆	-4,552⋆	-6,685⋆
F_5	-0,851	-0,657	-1,066	18,875 ⋆	11,48⋆	-2,021⋆
F_6	-23,67⋆	-27,96⋆	-17,37⋆	-18,672 ⋆	-26,34⋆	-18,86⋆
F_7	-19,78⋆	-7,367⋆	-15,80⋆	-7,263⋆	-5,029⋆	-13,96⋆
F_8	10,90⋆	9,827⋆	-8,315⋆	-1,959 ⋆	-9,168⋆	1,745⋆
F_9	-85,03⋆	-79,82⋆	-44,87⋆	-15,762⋆	-24,58⋆	-43,86⋆
F_{10}	-63,55⋆	-90,98⋆	-45,54⋆	-14,452⋆	-21,68⋆	-33,88⋆
F_{11}	-19,63⋆	-18,21⋆	-18,71⋆	-8,565⋆	-11,57⋆	-18,37⋆
F_{12}	-3,300⋆	8,242⋆	-9,374⋆	13,937⋆	-5,776⋆	-7,239⋆
F_{13}	27,86⋆	27,88⋆	2,339⋆	24,416⋆	4,357⋆	27,91⋆
F_{14}	-0,445	1,000	-1,368	-2,997⋆	-3,344⋆	-1,73⋆
F_{15}	-1,514	-1,928⋆	-0,758	-1,066	-2,243⋆	-1,379
F_{16}	-18,11⋆	-18,11⋆	-18,11⋆	-18,117⋆	-2,490⋆	1,291
F_{17}	-31,27⋆	-31,27⋆	-31,27⋆	-23,273⋆	-5,915⋆	4,238⋆
F_{18}	2,052⋆	2,052⋆	2,052⋆	1,679 ⋆	0,629	2,052⋆
F_{19}	1,903⋆	1,903⋆	-4,131⋆	-2,460⋆	-11,60⋆	1,739⋆
F_{20}	3,053⋆	1,757⋆	0,050	1,588	-5,190⋆	-0,933
F_{21}	-3,257⋆	-5,192⋆	-2,576⋆	-0,644	-21,01⋆	-1,661
F_{22}	-3,857⋆	-3,502⋆	-2,281⋆	3,460⋆	-20,50⋆	-1,636
F_{23}	-3,269⋆	-2,368⋆	-0,947	0,441	-19,82⋆	-2,625⋆

IV.7 Application de LBPO en BM

Dans le cadre d'estimation de mouvement, une nouvelle technique de BM basée sur la méthode LBPO appelée LBPO-BM est proposée. Cette dernière est évaluée sur six séquences vidéo et les résultats en termes de D_{psnr} et de complexité de calcul sont comparés avec différentes techniques BM : FSA, TSS, NTSS, SES, 4SS, DS, MPBM, SDDS, PSO-BM, BH-BM, SCA-BM, GWO-BM, SSA-BM, MS-BM, PLA-BM et TPLA-BM et notre méthode SFS-BM présentée dans le chapitre précédent.

Les principales étapes de la méthode LBPO-BM sont :

1. Initialisation des blocs candidats dans la fenêtre de recherche.

2. Les blocs candidats sont évalués avec une fonction de fitness (Eq. (IV.20)).

$$\boldsymbol{F} = \sum_{i=1}^{B_x} \sum_{j=1}^{B_y} |C(x+i,y+j) - R(x+i+u,y+j+v)| \tag{IV.20}$$

3. Pour chaque bloc candidat, on génère un code spécifique par l'équation IV.8, puis on calcule une valeur de probabilité par l'équation IV.12.

4. Selon ces valeurs de probabilité, les positions des blocs candidats sont mises à jour à l'aide de l'équation IV.13.

5. Les processus de LBPO sont répétés jusqu'au nombre maximal d'itérations.

L'algorithme 8 donne un pseudo-code de la méthode proposée LBPO-BM.

Initialisation de la taille de la population N_p,
Initialisation du nombre maximal d'itérations,
Initialisation de la taille de la fenêtre de recherche w_{max},
Initialisation des tailles de bloc $B_x \times B_y$,
Lire la séquence vidéo I,
Pour t **de** 1 **à** le nombre maximal d'images **faire**
  Image courante = $I(t)$
  Image référence = $I(t-1)$ ou $I(t+1)$
  Diviser l'image courante en blocs
  Déterminer les blocs statiques
  Pour $iter$ **de** 1 **à** le nombre maximal d'itérations **faire**
    Évaluation des blocs candidats par la fonction de fitness (IV.20)
    Générer les codes par l'équation IV.8
    Calculer les valeurs de probabilités par l'équation IV.12
    Mise à jour de la population en utilisant l'équation IV.13
  Fin Pour
  Les vecteurs de mouvements sont calculés en utilisant les meilleurs blocs trouvés.
Fin Pour

Algorithme 8 : Algorithme LBPO-BM

Les résultats obtenus avec notre LBPO-BM et les autres méthodes de BM sont présentés dans les tableaux IV.8 et IV.9.

En terme de complexité de calcul, le tableau IV.8 montre que notre algorithme LBPO-BM évalue un nombre de bloc, égal à 9.98, le plus réduit par rapport aux autres méthodes. Ce qui prouve que l'algorithme LBPO-BM procure un processus de recherche très efficace. Notre méthode LBPO-BM est meilleure que la méthode FSA, en temps de calcul, avec un pourcentage d'amélioration égale à 85.45%.

En termes de précision d'estimation de mouvement, le tableau IV.9 montre que notre algorithme LBPO-BM donne une valeur moyenne de D_{psnr} très petite égale à $-0,56$. Malgré que la valeur de D_{psnr} obtenue avec LBPO-BM est supérieure à celle obtenue avec SFS-BM, mais la méthode LBPO-BM est meilleure que la méthode SFS-BM en terme de complexité de calcul, ce qui représente un échange intéressant entre eux.

D'après ces résultats, l'algorithme LBPO-BM peut assurer un bon compromis entre la

précision et la complexité de calcul. Les résultats confirment également la capacité de la méthode LBPO à résoudre des problèmes concrets et produire des résultats compétitifs par rapport aux autres méthodes récentes.

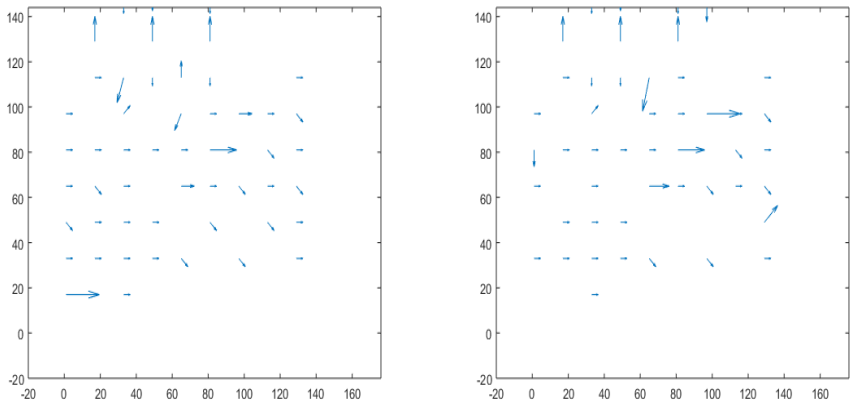
TABLE IV.8. : Le nombre de blocs évalués par les différents algorithmes BM

Algorithme		Akiyo	Container	Carphone	Foreman	Hall	Stefan	Moyenne	Classement
FSA	(1981)	236,63	236,63	236,63	236,63	262,62	984,91	365,67	18
TSS	(1994)	21,48	25	25	25	23,25	25	24,12	16
NTSS	(1994)	14,68	17,2	21,8	22,1	16,92	25,4	19,68	14
SES	(1997)	16,2	19,2	18,86	18,85	16,92	17,45	17,91	12
4SS	(1996)	14,65	19	25,25	24,8	16,25	25,3	20,87	15
DS	(2000)	11,43	7,5	12,5	13,4	12,89	15,2	12,15	6
MPBM	(2011)	10.04	5.91	11.22	11.49	11.06	11.64	10.23	3
SDDS	(2017)	11.45	11.42	13.06	14.83	12.88	17.25	13.48	7
PSO-BM	(1995)	12,60	6,11	13,32	13,72	12,02	14,77	12.09	5
BH-BM	(2013)	15,18	8,04	15,20	15,25	15,22	16,25	14.19	8
SCA-BM	(2016)	19,79	10,42	19,79	19,79	20,01	20,65	18.41	13
GWO-BM	(2014)	15,65	8,40	15,60	15,66	15,95	16,55	14.64	9
SSA-BM	(2017)	16,61	8,79	16,77	17,02	17,04	19,20	15.90	11
MS-BM	(2018)	16,04	8,53	16,02	16,03	16,16	16,58	14.89	10
PLA-BM	(2016)	49,7	75,27	71,54	77,5	//	131,7	81,14	17
TPLA-BM	(2016)	8,69	10,85	9,17	10,09	//	13,9	10,54	4
SFS-BM	(2019)	1,41	1,41	7,52	14,62	15,45	19,65	10,01	2
LBPO-BM	(2019)	5,37	10,19	9,45	9,98	10,26	14,63	**9,98**	1

TABLE IV.9. : Les valeurs de D_{psnr} obtenues avec les différents algorithmes BM

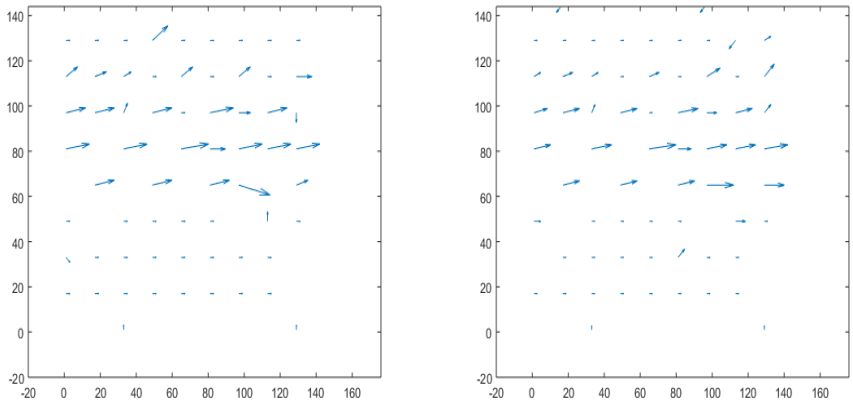
Algorithme		Akiyo	Container	Carphone	Foreman	Hall	Stefan	Moyenne	Classement
FSA	(1981)	0	0	0	0	0	0	0	1
TSS	(1994)	0	-0,2	-3,92	-7,32	-0,14	-18,52	-5,01	17
NTSS	(1994)	0	-0,15	-3,67	-3,57	-0,11	-13,2	-3,45	10
4SS	(1996)	0	-0,15	-4,01	-7,44	-0,14	-17,48	-4,87	14
SES	(1997)	0	-0,15	-4,5	-7,73	-0,25	-17,64	-5,04	18
DS	(2000)	0	-0,13	-0,79	-1,59	-0,14	-3,73	-1,06	8
MPBM	(2011)	-0.14	0.01	-0.77	-0.68	-0.15	-1.58	-0.55	4
SDDS	(2017)	-0.08	0.01	-0.62	-1.14	-0.11	-3.16	-0.85	6
PSO-BM	(1995)	-0,25	0,01	-1,58	-4,28	-0,32	-10,88	-2.88	9
BH-BM	(2013)	-0,11	-0,52	-3,05	-6,12	-1,20	-17,42	-4.74	13
GWO-BM	(2014)	-0,10	-0,37	-3,02	-5,78	-1,11	-17,09	-4.58	12
SCA-BM	(2016)	-0,09	-0,72	-3,44	-6,09	-1,35	-18,04	-4.95	16
SSA-BM	(2017)	-0,12	-0,58	-2,90	-5,49	-1,10	-15,86	-4.34	11
MS-BM	(2018)	-0,12	-0,64	-3,28	-6,25	-1,31	-17,76	-4.89	15
PLA-ME	(2016)	0	0	-0,28	-0,82	//	-3,19	-0,85	7
TPLA-ME	(2016)	0	0	-0,28	-0,85	//	-2,92	-0,81	5
SFS-BM	(2019)	-0,02	0	0	0	-0,02	-2,04	**-0,34**	2
LBPO-BM	(2019)	0	-0,02	-1,04	-1,16	-0,2	-0,96	-0,56	3

Les figures IV.13 et IV.14 illustrent les vecteurs de mouvement obtenus avec la méthode FSA et ceux obtenus avec notre méthode LBPO-BM, dans les séquences vidéos Carphone et Foreman. Nous remarquons que les résultats de FSA et de LBPO-BM ont la même allure.



(a) FSA (b) LBPO-BM

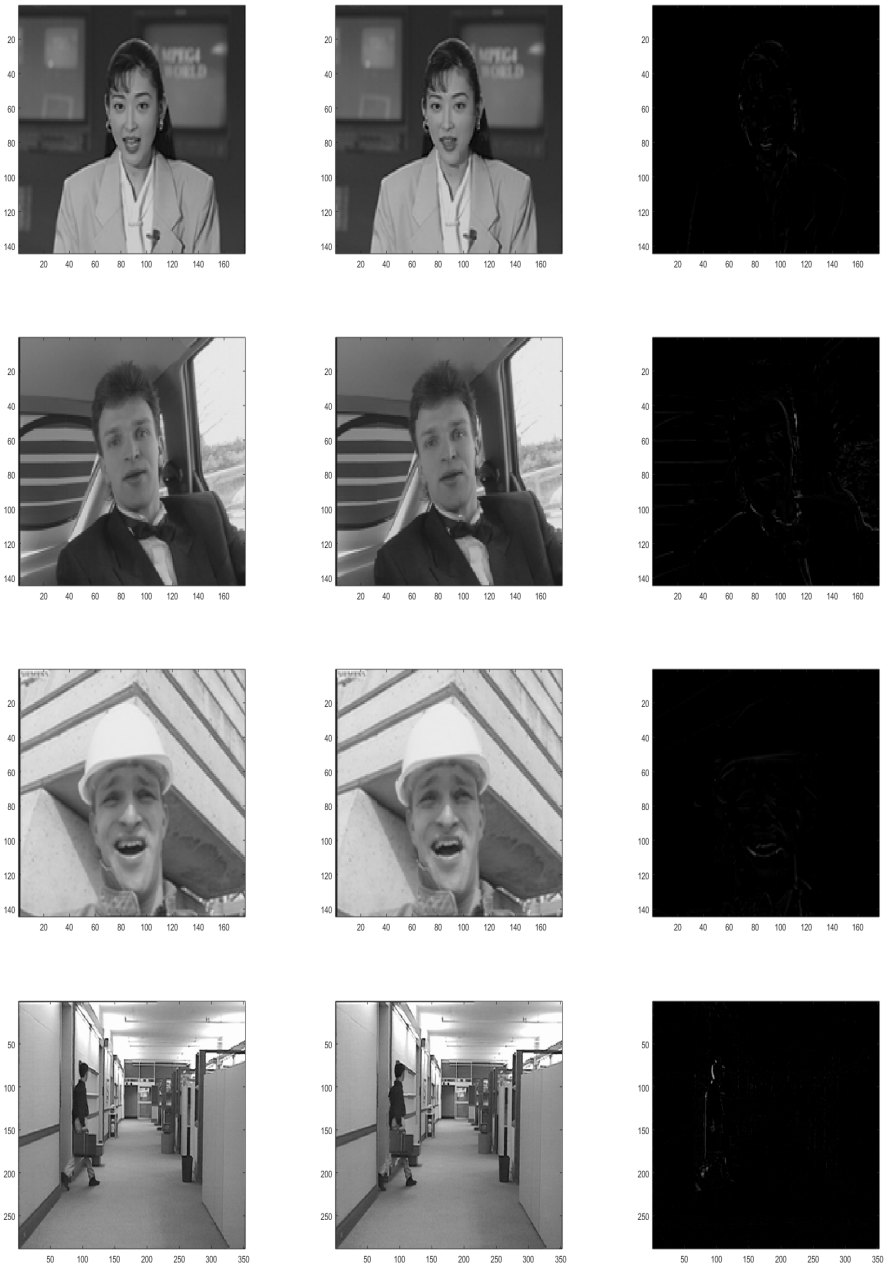
FIGURE IV.13. : Résultats sur la séquences vidéo Carphone.



(a) FSA (b) LBPO-BM

FIGURE IV.14. : Résultats sur la séquences vidéo Foreman.

La figure IV.15 illustrent dans la première colonne l'image courante, dans la deuxième colonne l'image compensée ou reconstruite et dans la troisième colonne l'image erreur obtenue par notre méthode LBPO-BM, sur différentes séquences vidéos. Nous remarquons que l'image reconstruite est très proche de l'image courante, mais certains déformations apparaissent au niveau des images reconstruites. Nous pouvons améliorer les résultats de la méthode en utilisant les idées proposées par SFS-BM dans le chapitre précédent.



(j) Image courante (k) Image compensée (l) Image erreur

FIGURE IV.15. : Résultats de l'algorithme LBPO-BM sur les séquences Akiyo, Carphone, Foreman et Hall.

IV.8 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode métaheuristique d'optimisation appelée LBPO inspirée du concept de base de l'opérateur LBP. L'idée nouvelle de cette méthode est le choix des bonnes solutions utilisées pour guider l'évolution de la population ne dépend pas des valeurs de fitness, mais sur le code donné à chaque solution. L'adoption de cette stratégie a permis l'identification des bonnes solutions dans chaque région dans l'espace de recherche qui laisse à l'algorithme la possibilité d'explorer plusieurs régions avant la convergence vers la meilleure solution globale.
Notre méthode a été évaluée en utilisant différentes fonctions de test. Les résultats obtenus par LBPO ont été comparés à ceux obtenus par d'autres métaheuristiques. Les expériences ont montré qu'aucune méthode métaheuristique ne surpasse complètement les autres, mais la méthode proposée LBPO obtient les meilleurs résultats pour la majorité des fonctions de test. En outre, nous avons montré que LBPO a des meilleures performances dans les problèmes de très hautes dimensions.
Dans le cadre de l'estimation de mouvement, la technique LBPO a été utilisé pour résoudre le problème de BM, les résultats obtenus pour des différentes séquences vidéos montrent que cette méthode offre des performances compétitives en termes de précision et de complexité de calcul par rapport aux autres méthodes utilisées pour la comparaison.

Conclusion générale

L'objectif de cette thèse était l'introduction des techniques d'optimisation métaheuristiques dans le domaine de l'estimation de mouvement. Les motivations des choix des méthodes simulées sont les suivantes : d'une part, les méthodes de mise en correspondance de blocs ou block matching (BM) sont les méthodes d'estimation de mouvement les plus utilisées grâce à leur implémentation simple et efficace. Les performances de ce type de méthodes en termes de précision et complexité de calcul dépendent de la stratégie de recherche adoptée. D'autre part, les techniques d'optimisation métaheuristiques représentent des outils permettant la résolution satisfaisante d'un grand nombre de problèmes d'optimisation avec une complexité de calcul réduite. De ce fait, nous avons développé dans notre travail des méthodes de BM basées sur les métaheuristiques.

Dans cette thèse, deux méthodes ont vu le jour :

La première méthode proposée SFS-BM consiste à résoudre le problème de BM en utilisant la technique de recherche fractale stochastique (SFS), une implémentation parallèle est utilisée pour estimer les mouvements de tous les blocs simultanément. De nouvelles idées concernant l'initialisation, la fonction de fitness, la fenêtre de recherche et l'approximation de la fonction de fitness sont également proposées pour encore plus améliorer les résultats. La technique métaheuristique SFS a retenu notre attention pour ses capacités à résoudre les problèmes d'optimisation difficiles et permet d'obtenir des résultats de bonne qualité. La méthode SFS-BM a été testée sur différentes séquences vidéos très connues. Les résultats de simulation obtenus montrent la supériorité de cette méthode par rapport aux autres techniques BM en termes de précision d'estimation de mouvement et en complexité de calcul.

Cependant, le problème majeur de la technique SFS et de toutes les techniques évo-

lutionnaires est le nombre d'évaluations de la fonction de fitness qui est élevé relativement aux autres techniques métaheuristiques, puisqu'elles utilisent plusieurs processus de mise à jour. La deuxième contribution consiste à développer une nouvelle technique méteheuristique capable à résoudre le problème de BM et donner un bon compromis entre la précision d'estimation de mouvement et la complexité de calcul.

LBP est un opérateur très connu en traitement d'image, il permet d'analyser la texture et identifier les détails dans les images. Une nouvelle méthode métaheuristique nommée optimisation à base de LBP (local binary pattern optimizer : LBPO) inspirée par le concept de base de l'opérateur LBP est proposée. LBPO a été évaluée avec des fonctions de test puis appliquée en BM. D'après les résultats obtenus aucune méthode métaheuristique ne surpasse complètement les autres, mais la méthode proposée LBPO a donné des résultats meilleurs que d'autres techniques métaheuristiques, comme PSO, CA, BH, GWO, SCA et SSA, pour la majorité des fonctions de test utilisées.
Les résultats d'application montrent que la méthode LBPO-BM procure des résultats satisfaisants qui dépassent d'autres technique BM connues dans la littérature.

En comparant les résultats d'estimation de mouvement obtenus avec la technique SFS-BM et la technique LBPO-BM, nous remarquons que les performances de LBPO-BM sont supérieures aux celles de la technique SFS-BM en terme de complexité de calcul et la méthode SFS-BM est meilleure que la méthode LBPO-BM en terme de précision, ce qui représente un échange intéressant entre eux.

Perspectives

Comme perspectives, nous proposons l'utilisation de la méthode SFS-BM dans des applications diverses comme la compression vidéo, le suivi d'objet, la surveillance, la robotique, etc.

La méthode LBPO peut être analysée expérimentalement d'une manière plus poussée, ensuite étendue en version binaire ou multi-objective.

Nous proposons aussi la combinaison de la technique LBPO avec d'autres méthodes d'optimisation exactes ou métaheuristiques.

Des travaux futurs pourraient s'intéresser à l'exploitation de la technique LBPO pour résoudre d'autres problèmes d'optimisation dans différents domaines : médical, industrie, communication, cryptage, etc.

Bibliographie

[1] B. KP. Horn & B. G. Schunck. “Determining optical flow”. *Artificial intelligence*, 1981, vol. 17, n. 1-3, p. 185-203.

[2] B. D. Lucas & T. Kanade. “An iterative image registration technique with an application to stereo vision”. *In Proc. Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada, 1981, p. 674-679.

[3] T. Brox, A. Bruhn, N. Papenberg, *et al.* “High accuracy optical flow estimation based on a theory for warping”. *European conference on computer vision*. Springer 2004, p. 25-36.

[4] P. Krähenbühl & V. Koltun. “Efficient nonlocal regularization for optical flow”. *European Conference on Computer Vision*. Springer 2012, p. 356-369.

[5] J. Fan, Y. Wang & L. Guo. “Hierarchical coherency sensitive hashing and interpolation with RANSAC for large displacement optical flow”. *Computer Vision and Image Understanding*, 2018, vol. 175, p. 1-10.

[6] M. Khalid, L. Pénard & E. Mémin. “Optical flow for image-based river velocity estimation”. *Flow Measurement and Instrumentation*, 2019, vol. 65, p. 110 - 121, issn. 0955-5986.

[7] C. H. Hsieh & T. P. Lin. “VLSI architecture for block-matching motion estimation algorithm”. *IEEE Transactions on Circuits and Systems for Video Technology*, 1992, vol. 2, n. 2, p. 169-175.

[8] J. Jain & A. Jain. “Displacement measurement and its application in interframe image coding”. *IEEE Transactions on communications*, 1981, vol. 29, n. 12, p. 1799-1808.

[9] H. H. Jong, L. G. Chen & T. D. Chiueh. “Accuracy improvement and cost reduction of 3-step search block matching algorithm for video coding”. *IEEE Transactions on Circuits and Systems for Video Technology*, 1994, vol. 4, n. 1, p. 88-90.

[10] N. Raj & S. Shabeer. “Hexagonal Search Based Compression Noise Estimation and Reduction in Video”. *Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. IEEE 2018, p. 1061-1064.

[11] J. Cai, & W. D. Pan. “On fast and accurate block-based motion estimation algorithms using particle swarm optimization”. *Information Sciences*, 2012, vol. 197, p. 53-64.

[12] M. A. Diaz-Cortés, E. Cuevas & R. Rojas. “Motion Estimation Algorithm Using Block-Matching and Harmony Search Optimization”. Engineering Applications of Soft Computing, 2017, p. 13-44. *Springer*.

[13] K. Bhattacharjee, S. Kumar, H. M. Pandey, *et al.* “An improved block matching algorithm for motion estimation in video sequences and application in robotics”. *Computers & Electrical Engineering*, 2018, vol. 68, p. 92-106.

[14] A. Puri, H. M. Hang & D. Schilling. “An efficient block-matching algorithm for motion-compensated coding”. *ICASSP’87, International Conference on Acoustics, Speech, and Signal Processing*. IEEE’1987, vol. 12, p. 1063-1066.

[15] S. Zhu. “A new diamond search algorithm for fast block-matching motion estimation”. *IEEE Transaction on Image Processing*, 2000, vol. 9, n. 2, p. 287-290.

[16] W. Li & E. Salari. “Successive elimination algorithm for motion estimation”. *IEEE Transactions on Image Processing*, 1995, vol. 4, n. 1, p. 105-107.

[17] H. C. Huang, Y. P. Hung & W. L. Hwang. “Adaptive early jump-out technique for fast motion estimation in video coding”. *Proceedings of the 13th International Conference on Pattern Recognition*. IEEE’1996, vol. 2, p. 864-868.

[18] S. I. A. Pandian, G. J. Bala & J. Anitha. “A pattern based PSO approach for block matching in motion estimation”. *Engineering Applications of Artificial Intelligence*, 2013, vol. 26, n. 8, p. 1811-1817.

[19] E. Cuevas, D. Zaldivar, M. Pérez-Cisneros, *et al.* “Block matching algorithm for motion estimation based on Artificial Bee Colony (ABC)”. *journal*, 2013, vol. 13, n. 6, p. 3047-3059.

[20] E. Cuevas, D. Zaldivar, M. Pérez-Cisneros, *et al.* “Block-matching algorithm based on differential evolution for motion estimation”. *Engineering Applications of Artificial Intelligence*, 2013, vol. 26, n. 1, p. 488-498.

[21] K. Hussain, M. N. Salleh, S. Cheng, *et al.* “Metaheuristic research : a comprehensive survey”. *Artificial Intelligence Review*, 2018, p. 1-43.

[22] F. Cherif. “Estimation hiérarchique du mouvement par ondelettes géométriques”. *Thèse doctorat : Électronique, l’université Mohamed Khider de Biskra,* 2015.

[23] M. Etienne. “Estimation du flot-optique : contributions et panorama de différentes approches”. *Habilitation à diriger des recherches, l’université Rennes 1 Institut de Formation Supérieure en Informatique et en Communication*, 2003.

[24] C. Grava. “Compensation de mouvement par réseaux neuronaux cellulaires. Application en imagerie médicale”. *Thèse doctorat : Images et Systèmes, l’université INSA de Lyon*, 2003.

[25] A. Marion. “Filtrage spatiotemporel orienté de séquences d’images : application à l’estimation du mouvement des flux sanguins en imagerie ultrasonore”. *Thèse doctorat : Acoustique et traitement de signal/image, l’université INSA de Lyon*, 2009.

[26] D. Fortun, P. Bouthemy & C. Kervrann. “Optical flow modeling and computation : a survey”. *Computer Vision and Image Understanding*, 2015, vol. 134, p. 1-21.

[27] P. Brault. “Estimation de mouvement et segmentation d’image”. *Thèse doctorat : Élec-*

tronique, l'université Paris-Sud XI, école doctorale sciences et technologies de l'information des télécommunications et des systèmes Paris-Sud XI, 2005.

[28] N. Laveau. "Mouvement et vidéo : estimation, compression et filtrage morphologique". *Thèse doctorat : Mathematics, l'école Nationale Supérieure des Mines de Paris*, 2005.

[29] C. Bernard. "Ondelettes et problèmes mal posés : la mesure du flot optique et l'interpolation irrégulière". *Mémoire, l'école Polytechnique, Palaiseau*, 1999.

[30] P. Weinzaepfel. "Motion in action : optical flow estimation and action localization in videos". *Thèse doctorat : Mathématiques, Sciences et Technologies de l'Information, l'université Grenoble Alpes*, 2016.

[31] S. S. Beauchemin & J. L. Barron. "The computation of optical flow". *ACM computing surveys (CSUR)*, 1995, vol. 27, n. 3, p. 433-466.

[32] A. Wedel & D. Cremers. "Stereo scene flow for 3D motion analysis". 2011. *Springer Science and Business Media*.

[33] M. Fradet. "Contributions à la segmentation de séquences d'images au sens du mouvement dans un contexte semi-automatique". *Thèse doctorat : Interface homme-machine, l'université Rennes 1*, 2010.

[34] M. Jakubowski, G. Pastuszak. "Block-based motion estimation algorithms a survey". *Opto-Electronics Review*, 2013, vol. 21, n. 1, p. 86-102.

[35] A. Geiger, P. Lenz & R. Urtasun. "Are we ready for autonomous driving ? the Kitti vision benchmark suite". *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE'2012, p. 3354-3361.

[36] H. Chao, Y. Gu & M. Napolitano. "A survey of optical flow techniques for robotics navigation applications". *Journal of Intelligent and Robotic Systems*, 2014, vol. 73, n. 1-4, p. 361-372.

[37] W. Enkelmann. “Obstacle detection by evaluation of optical flow fields from image sequences”. *Image and Vision Computing*, 1991, vol. 9, n. 3, p. 160-168.

[38] A. Basset, P. Bouthemy & C. Kervrann. “Recovery of motion patterns and dominant paths in videos of crowded scenes ”. *ICIP*. 2014, p. 184-188.

[39] M. J. Black & Y. Yacoob. “Recognizing facial expressions in image sequences using local parameterized models of image motion”. *International Journal of Computer Vision*, 1997, vol. 25, n. 1, p. 23-48.

[40] N. Kiryati, T. R. Raviv, Y. Ivanchenko, *et al.* “Real-time abnormal motion detection in surveillance video”. *19th International Conference on Pattern Recognition ICPR*. IEEE’2008, p. 1-4.

[41] T. C. Huang, C. K. Chang, C. H. Liao, *et al.* “Quantification of blood flow in internal cerebral artery by optical flow method on digital subtraction angiography in comparison with time-of-flight magnetic resonance angiography”. *PloS one*, 2013, vol. 8, n. 1, p. e54678.

[42] M. Xavier, A. Lalande, P. M. Walker, *et al.* “An adapted optical flow algorithm for robust quantification of cardiac wall motion from standard cine-mr examinations”. *IEEE Transactions on Information Technology in Biomedicine*, 2012, vol. 16, n. 5, p. 859-868.

[43] N. Hata, A. Nabavi, W. Wells III, *et al.* “Three-dimensional optical flow method for measurement of volumetric brain deformation from intraoperative MR images”. *Journal of Computer Assisted Tomography*, 2000, vol. 24, n. 4, p. 531-538.

[44] S. F. Gissot, J. F. Hochedez, P. Chainais, *et al.* “3D reconstruction from SECCHI-EUVI images using an optical-flow algorithm : method description and observation of an erupting filament”. *Solar Physics*, 2008, vol. 252, n. 2, p. 397-408.

[45] M. Werlberger, T. Pock, M. Unger, *et al.* “Optical flow guided TV-L 1 video interpolation and restoration”. *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer’2011, p. 273-286.

[46] Q. Yao, Q. Liu, T. G. Dietterich, *et al.* "Segmentation of touching insects based on optical flow and NCuts". *biosystems engineering*, 2013, vol. 114, n. 2, p. 67-77.

[47] T. Corpetti, M. étienne & P. Pérez. "Dense estimation of fluid flows". *IEEE Transactions on pattern analysis and machine intelligence*, 2002, vol. 24, n. 3, p. 365-380.

[48] P. Héas, M. Etienne, N. Papadakis. "Layered estimation of atmospheric mesoscale dynamics from satellite imagery". *IEEE Transactions on Geoscience and Remote Sensing*, 2007, vol. 45, n. 12, p. 4087-4104.

[49] T. Liu, L. Shen. "Fluid flow and optical flow". *Journal of Fluid Mechanics*, 2008, vol. 614, p. 253-291.

[50] J. L. Barron, D. J. Fleet & S. S. Beauchemin "Performance of optical flow techniques". *International Journal of Computer Vision*, 1994, vol. 12, n. 1, p. 43-77.

[51] B. Galvin, B. McCane, K. Novins, *et al.* "Recovering Motion Fields : An Evaluation of Eight Optical Flow Algorithms". *British Machine Vision Conference*, 1998, vol. 98, p. 195-204.

[52] S. Baker, D. Scharstein, J.P. Lewis, *et al.* "A database and evaluation methodology for optical flow". *International Journal of Computer Vision*, 2011, vol. 92, n. 1, p. 1-31.

[53] D. Sun, S. Roth & M. J. Black. "Secrets of optical flow estimation and their principles". *IEEE' 2010 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, p. 2432-2439.

[54] M. J. Black & P. Anandan. "The robust estimation of multiple motions : Parametric and piecewise-smooth flow fields". *Computer Vision and Image Understanding*, 1996, vol. 63, n. 1, p. 75-104.

[55] M. J. Black, Y. Yacoob, A. D. Jepson, *et al.* "Learning parameterized models of image motion". *Proceedings of the Conference on Computer Vision and Pattern Recognition, Computer Society*. IEEE 1997, p. 561-567.

[56] K. Karantzalos & N. Paragios. “Higher order polynomials, free form deformations and optical flow estimation”. *International Conference on Image Processing (ICIP)*. IEEE 2005, vol. 3, p. III-1280.

[57] J. M. Odobez & P. Bouthemy. “Robust multiresolution estimation of parametric motion models”. *Journal of visual communication and image representation*, 1995, vol. 6, n. 4, p. 348-365.

[58] N. Papenberg, A. Bruhn, T. Brox, *et al.* “Highly accurate optic flow computation with theoretically justified warping”. *International Journal of Computer Vision*, 2006, vol. 67, n. 2, p. 141-158.

[59] A. Wedel, D. Cremers, T. Pock, *et al.* “Structure-and motion-adaptive regularization for high accuracy optic flow”. *12th International Conference on Computer Vision*. IEEE’2009, p. 1663-1668.

[60] P. Golland & A. M. Bruckstein. “Motion from color”. *Computer Vision and Image Understanding*, 1997, vol. 68, n. 3, p. 346-362.

[61] J. Weber & J. Malik. “Robust computation of optical flow in a multi-scale differential framework”. *International Journal of Computer Vision*, 1995, vol. 14, n. 1, p. 67-81.

[62] T. Hyun Kim, H. Seok Lee & K. Mu Lee. “Optical flow via locally adaptive fusion of complementary data costs”. *Proceedings of the IEEE International Conference on Computer Vision*. IEEE’2013, p. 3344-3351.

[63] D. Shulman & J. Y. Herve. “Regularization of discontinuous flow fields”. *Proceedings, Workshop on Visual Motion*. IEEE’1989, p. 81-86.

[64] W. Trobin, T. Pock, D. Cremers, *et al.* “An unbiased second-order prior for high-accuracy motion estimation”. Joint Pattern Recognition Symposium, 2008, p. 396-405. *Springer*.

[65] F. Heitz & P. Bouthemy. “Multimodal estimation of discontinuous optical flow using

Markov random fields". *IEEE Transactions on pattern analysis and machine intelligence*, 1993, vol. 15, n. 12, p. 1217-1232.

[66] T. M. Chin, W. C. Karl & A. S. Willsky. "Probabilistic and sequential computation of optical flow using temporal coherence". *IEEE Transactions on Image Processing*, 1994, vol. 3, n. 6, p. 773-788.

[67] V. Kolmogorov & R. Zabih. "Computing visual correspondence with occlusions using graph cuts". *Proceedings. Eighth IEEE International Conference on Computer Vision, (ICCV)*. IEEE'2001, vol. 2, p. 508-515.

[68] A. Ayvaci, M. Raptis & S. Soatto. "Sparse occlusion detection with optical flow". *International Journal of Computer Vision*, 2012, vol. 97, n. 3, p. 322-338.

[69] M. Leordeanu, A. Zanfir & C. Sminchisescu. "Locally affine sparse-to-dense matching for motion and occlusion estimation". *Proceedings of the IEEE International Conference on Computer Vision*. IEEE'2013, p. 1721-1728.

[70] Y. Hu, Y. Li & R. Song. "Robust interpolation of correspondences for large displacement optical flow". *Proceedingsof the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE'2017.

[71] D. Sun, X. Yang, M. Y. Liu, *et al.* "Pwc-net : Cnns for optical flow using pyramid, warping, and cost volume". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE'2018, p. 8934-8943.

[72] R. Schuster, O. Wasenmuller, G. Kuschk, *et al.* "Scene flow fields : Dense interpolation of sparse scene flow correspondences". *Winter Conference on Applications of Computer Vision (WACV)*. IEEE'2018, p. 1056-1065.

[73] D. H. Trinh & C. Daul. "On illumination-invariant variational optical flow for weakly textured scenes". *Computer Vision and Image Understanding*, 2018, doi : https ://doi.org/10.1016/j.cviu.2018.11.004.

[74] M. Zarei-Jalalabadi, S. M. B. Malaek. "Motion estimation of uncooperative space ob-

jects : A case of multi-platform fusion". *Advances in Space Research*, 2018, vol. 62, n. 9, p. 2665-2678.

[75] T. Zhigang, X. Wei, Z. Dejun, *et al.* "A survey of variational and CNN-based optical flow techniques". *Signal Processing : Image Communication*, 2019, vol. 72, p. 9-24, issn. 0923-5965.

[76] D. Fleet & Y. Weiss. "Optical flow estimation". Handbook of mathematical models in computer vision, 2006, p. 237-257. *Springer*.

[77] C. Braillon. "Détection d'obstacles par fusion de flux optique et stéréovision dans des grilles d'occupation". *Thèse doctorat : Informatique, Institut National Polytechnique de Grenoble*, 2007.

[78] T. Ensem, W. Didier & A. Ducrot. "Estimation temps réel du Flot Optique". *Rapport de stage, INRIA, ISA*. 2007.

[79] A. Basarab. "Estimation du mouvement dans des séquences d'images échographiques : application à l'élastographie de la thyroide". *Thèse doctorat : Images et Systèmes, INSA-Lyon*, 2008.

[80] W. R. Crum, T. Hartkens & D. Hill. "Non-rigid image registration : theory and practice". *The British journal of radiology*, 2004, vol. 77, n. suppl2, p. S140-S153.

[81] C. Nastar. "Modèles physiques déformables et modes vibratoires pour l'analyse du mouvement non-rigide dans les images multidimensionnelles". *Thèse doctorat : Modélisation et simulation, École Nationale des Ponts et Chaussées*, 1994.

[82] C. Vogel, S. Roth & K. Schindler. "An evaluation of data costs for optical flow". *German Conference on Pattern Recognition*. Springer 2013, p. 343-353.

[83] S. Uras, F. Girosi, A. Verri, *et al.* "A computational approach to motion perception". *Biological Cybernetics*, 1988, vol. 60, n. 2, p. 79-87.

[84] L. G. Brown. “A survey of image registration techniques”. *ACM computing surveys (CSUR)*, 1992, vol. 24, n. 4, p. 325-376.

[85] H. Zimmer, A. Bruhn & J. Weickert. “Optic flow in harmony”. *International Journal of Computer Vision*, 2011, vol. 93, n. 3, p. 368-388.

[86] J. F. Aujol, G. Gilboa, T. Chan, *et al.* “Structure-texture image decomposition-modeling, algorithms, and parameter selection”. *International Journal of Computer Vision*, 2006, vol. 67, n. 1, p. 111-136.

[87] J. van de Weijer & T. Gevers. “Robust optical flow from photometric invariants”. *ICIP’04, International Conference on Image Processing*. IEEE 2004, vol. 3, p. 1835-1838.

[88] Y. Mileva, A. Bruhn & J. Weickert. “Illumination-robust variational optical flow with photometric invariants”. Joint Pattern Recognition Symposium, 2007, p. 152-162. *Springer*.

[89] A. Bruhn, J. Weickert & C. Schnörr. “Lucas/Kanade meets Horn/Schunck : Combining local and global optic flow methods”. *International Journal of Computer Vision*, 2005, vol. 61, n. 3, p. 211-231.

[90] M. Drulea & S. Nedevschi. “Total variation regularization of local-global optical flow”. *14th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE 2011, p. 318-323.

[91] L. Xu, J. Jia, & Y. Matsushita. “Motion detail preserving optical flow estimation”. *IEEE Transactions on pattern analysis and machine intelligence*, 2012, vol. 34, n. 9, p. 1744-1757.

[92] S. Negahdaripour. “Revised definition of optical flow : Integration of radiometric and geometric cues for dynamic scene analysis”. *IEEE Transactions on pattern analysis and machine intelligence*, 1998, vol. 20, n. 9, p. 961-979.

[93] M. Lefebure, L. Alvarez, J. Esclarin, *et al.* “A PDE model for computing the optical

flow". *Proceeding XVI congreso de ecuaciones diferenciales y aplicaciones*. 1999, p. 1349-1356.

[94] C. Schnörr. "Segmentation of visual motion by minimizing convex non-quadratic functionals". *Proceedings of the 12th IAPR International Conference on Pattern Recognition, 1-Conference A : Computer Vision and Image Processing*. IEEE'1994, vol. 1, p. 661-663.

[95] D. Sun, S. Roth, J. P. Lewis, *et al.* "Learning Optical Flow". Computer Vision - ECCV 2008, p. 83-97. *Springer Berlin Heidelberg*.

[96] H. H. Nagel & W. Enkelmann. "An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences". *IEEE Transactions on pattern analysis and machine intelligence*, 1986, n. 5, p. 565-593.

[97] J. Weickert & C. Schnörr. "A theoretical framework for convex regularizers in PDE-based computation of image motion". *International Journal of Computer Vision*, 2001, vol. 45, n. 3, p. 245-264.

[98] S. Roth, V. Lempitsky, C. Rother. "Discrete-continuous optimization for optical flow estimation". Statistical and Geometrical Approaches to Visual Motion Analysis, 2009, p. 1-22. *Springer*.

[99] M. Menze, C. Heipke & A. Geiger. "Discrete optimization for optical flow". *Conference on Pattern Recognition, German*. Springer'2015, p. 16-28.

[100] R. Fezzani. "Approche parallèle pour l'estimation du flot optique par méthode variationnelle". *Thèse doctorat : Traitement des images, l'université Pierre et Marie Curie-Paris VI*, 2011.

[101] C. Liu, J. Yuen & A. Torralba. "Sift flow : Dense correspondence across scenes and its applications". *IEEE Transactions on pattern analysis and machine intelligence*, 2011, vol. 33, n. 5, p. 978-994.

[102] T. Brox & J. Malik. "Large displacement optical flow : descriptor matching in varia-

tional motion estimation". *IEEE Transactions on pattern analysis and machine intelligence*, 2011, vol. 33, n. 3, p. 500-513.

[103] C. Barnes, E. Shechtman, D. B. Goldman, *et al.* "The generalized patchmatch correspondence algorithm". *European Conference on Computer Vision*. Springer'2010, p. 29-43.

[104] S. Metkar & S. Talbar. "Motion estimation techniques for digital video coding". 2013. *Springer*.

[105] I. Pirnog & C. C. Oprea. "Block Matching Motion Estimation with Variable Search Window Size". *International Journal on Advances in Software*, 2011, vol. 4, n. 3-4.

[106] "Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s-Part 2 : Video, ISO/IEC 11172-2". 1993.

[107] "Information Technology-Coding of Moving Pictures and Associated, Information Technology-Generic Coding of Moving Pictures and Associated Audio Information : Video, ISO/IEC 13818-2 and ITU-T Recommendation H.262". 1996.

[108] "Information Technology-Coding of Audio-Visual Objects-Part 2 : Visual, ISO/IEC 14496/2". 1999.

[109] "Video Codec for Audiovisual Services at p $\times$ 64 Kbit/s, ITU-T Recommendation H.261". 1993.

[110] "Video Coding for Low Bit Rate Communication, ITU-T Recommendation H.263". 1998.

[111] "Joint Video Team, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITUT Recommendation H.264 and ISO/IEC 14496/10 AVC". 2003.

[112] Y. Lai & L. Kun-lun. "Anti-occlusion target tracking based on improved block matching and AKF". *Transducer and Microsystem Technologies*, 2018, vol. 1, p. 014.

[113] J. Xiao, W. Zou, S. Zhang, *et al.* "Video denoising algorithm based on improved dual-domain filtering and 3D block matching". *IET Image Processing*, 2018, vol. 12, n. 12, p. 2250-2257.

[114] Z. Yong-xiang, W. Jian, Z. Wei-gong, *et al.* "An improved algorithm of electronic image stability based on block matching". *The 5th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE 2010, p. 1924-1927.

[115] F. Gamino-Sánchez, I. V. Hernández-Gutiérrez, A. J. Rosales-Silva, *et al.* "Block-Matching Fuzzy C-Means clustering algorithm for segmentation of color images degraded with Gaussian noise". *Engineering Applications of Artificial Intelligence*, 2018, vol. 73, p. 31-49.

[116] A. Nijad. "Fast Block Matching Criterion for Real-Time Video Communication". *International Conference on New Trends in Computing Sciences (ICTCS)*. IEEE 2017, p. 327-332.

[117] Y. Dazhi, W. M. Walsh, D. Zibo, *et al.* "Block matching algorithms : Their applications and limitations in solar irradiance forecasting". *Energy Procedia*, 2013, vol. 33, p. 335-342.

[118] K. Kaabneh, E. Abu-Hammad & F. Hamd. "Enhanced Skin Detection Technique Using Block Matching". *International Conference on Signal Processing and Communications (ICSPC)*. IEEE 2007, p. 21-24.

[119] M. U. Zoha. "Incorporating high frequency component matching motion estimation in H. 264 : A revolutionary improvement for real-time telemedicine applications". *14th International Conference on Computer and Information Technology (ICCIT)*. IEEE 2011, p. 509-513.

[120] L. M. Po & W. C. Ma. "A novel four-step search algorithm for fast block motion estimation". *IEEE Transactions on Circuits and Systems for Video Technology*, 1996, vol. 6, n. 3, p. 313-317.

[121] L. K. Liu & E. Feig. "A block-based gradient descent search algorithm for block mo-

tion estimation in video coding". *IEEE Transactions on Circuits and Systems for Video Technology*, 1996, vol. 6, n. 4, p. 419-422.

[122] Y. Nie & K. K. Ma. "Adaptive rood pattern search for fast block-matching motion estimation". *IEEE Transactions on image processing*, 2002, vol. 11, n. 12, p. 1442-1449.

[123] T. S. Shinde & A. K. Tiwari. "Efficient direction-oriented search algorithm for block motion estimation". *IET Image Processing*, 2018, vol. 12, p. 1567-1576, Issue, 9.

[124] J. Feng, K. T. Lo, H. Mehrpour, *et al.* "Adaptive block matching motion estimation algorithm using bit-plane matching". *Proceedings of the International Conference on Image Processing*. IEEE 1995, vol. 3, p. 496-499.

[125] S. Lee, J. M. Kim & S. I. Chae. "New motion estimation algorithm using adaptively quantized low bit-resolution image and its VLSI architecture for MPEG2 video encoding". *IEEE Transactions on Circuits and Systems for Video Technology*, 1998, vol. 8, n. 6, p. 734-744.

[126] B. Liu & A. Zaccarin. "New fast algorithms for the estimation of block motion vectors". *IEEE Transactions on Circuits and Systems for Video Technology*, 1993, vol. 3, n. 2, p. 148-157.

[127] C. K. Cheung & L. M. Po. "A hierarchical block matching algorithm using partial distortion measure". *ISCAS'97, Proceedings of the International Symposium on Circuits and Systems*. IEEE 1997, vol. 2, p. 1237-1240.

[128] K. M. Nam, J. S. Kim, R. H. Park, *et al.* "A fast hierarchical motion vector estimation algorithm using mean pyramid". *IEEE Transactions on Circuits and Systems for Video Technology*, 1995, vol. 5, n. 4, p. 344-351.

[129] J. Chalidabhongse & C. Kuo. "Fast motion vector estimation using multiresolution-spatio-temporal correlations". *IEEE Transactions on Circuits and Systems for Video Technology*, 1997, vol. 7, n. 3, p. 477-488.

[130] C. H. Lee & L. H. Chen. “A fast motion estimation algorithm based on the block sum pyramid”. *IEEE Transactions on Image Processing*, 1997, vol. 6, n. 11, p. 1587-1591.

[131] K. H. K. Chow & M. L. Liou. “Genetic motion search algorithm for video compression”. *IEEE Transactions on Circuits and Systems for Video Technology*, 1993, vol. 3, n. 6, p. 440-445.

[132] R. Ren, Y. Shi & B. Zheng. “A fast block matching algorithm for video motion estimation based on particle swarm optimization and motion prejudgment”. *arXiv preprint cs/0609131*, 2006.

[133] X. Yuan & X. Shen. “Block matching algorithm based on particle swarm optimization for motion estimation”. *ICESS’08, International Conference on Embedded Software and Systems*. IEEE’2008, p. 191-195.

[134] B. Dash & S. Rup. “An Improved Block-Matching Algorithm Based on Chaotic Sine-Cosine Algorithm for Motion Estimation”. *International Conference on Artificial Neural Networks*. Springer’2018, p. 759-770.

[135] C. M. Kuo, C. H. Hsieh, Y. D. Jou, *et al.* “Motion estimation for video compression using Kalman filtering”. *IEEE Transactions on Broadcasting*, 1996, vol. 42, n. 2, p. 110-116.

[136] P. Y. Chen & J. M. Jou. “An efficient blocking-matching algorithm based on fuzzy reasoning”. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 2001, vol. 31, n. 2, p. 253-259.

[137] F. Zhao, M. Zhou, G. Geng, *et al.* “Rigid blocks matching method based on contour curves and feature regions”. *IET Computer Vision*, 2017, vol. 12, n. 1, p. 76-85.

[138] W. T. Lin, T. Nanjundaswamy & K. Rose. “Adaptive Interpolated Motion-Compensated Prediction with Variable Block Partitioning”. *Conference on Data Compression*. IEEE 2018, p. 23-31.

[139] R. Li, B. Zeng & M. L. Liou. “A new three-step search algorithm for block motion

estimation". *IEEE Transactions on Circuits and Systems for Video Technology*, 1994, vol. 4, n. 4, p. 438-442.

[140] M. Ghanbari. "The cross-search algorithm for motion estimation (image coding)". *IEEE Transactions on communications*, 1990, vol. 38, n. 7, p. 950-953.

[141] A. M. Tourapis, O. c. L. Au & M. L. Liou. "Fast motion estimation using circular zonal search". *International Society for Optics and Photonics, Visual communications and image processing'99*. 1998, vol. 3653, p. 1496-1505.

[142] R. Srinivasan & K. Rao. "Predictive coding based on efficient motion estimation". *IEEE Transactions on communications*, 1985, vol. 33, n. 8, p. 888-896.

[143] J. Jain & A. Jain. "Displacement measurement and its application in interframe image coding". *IEEE Transactions on communications*, 1981, vol. 29, n. 12, p. 1799-1808.

[144] S. Goel, Y. Ismail & M. A. Bayoumi. "Adaptive search window size algorithm for fast motion estimation in H. 264/AVC standard". *48th Midwest Symposium on Circuits and Systems*. IEEE'2005, p. 1557-1560.

[145] N. Smairi. "Optimisation par essaim particulaire : adaptation de tribes à l'optimisation multiobjectif". *Thèse doctorat : Mathématiques, Sciences et Technologies de l'Information et de la Communication, l'université Paris Est*, 2013.

[146] S. S. Rao. "Engineering optimization : theory and practice". 2009. *John Wiley and Sons*.

[147] A. Betka. "Optimisation des dispositions de caméras en 2D par la technique PSO". *Mémoire de master : Électronique, l'université Biskra, Algérie*, 2015.

[148] A. El Dor. "Perfectionnement des algorithmes d'optimisation par essaim particulaire : applications en segmentation d'images et en électronique". *Thèse doctorat : Images, Signaux et Systèmes Intelligents, l'université Paris-Est*, 2012.

[149] H. Hachimi. "Hybridations d'algorithmes métaheuristiques en optimisation globale et

leurs applications". *Thèse doctorat : Optimisation et Fiabilité en Mécanique des Structures, l'université INSA de Rouen*, 2013.

[150] S. Abdesselam. "Conception d'un système d'optimisation pour le positionnnement de caméras pour la motion capture MOCAP". *Thèse doctorat : Électronique, l'université Biskra, Algérie*, 2018.

[151] I. Boussaid, J. Lepagnot & P. Siarry. "A survey on optimization metaheuristics". *Information Sciences*, 2013, vol. 237, p. 82-117.

[152] S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi. "Optimization by simulated annealing". *science*, 1983, vol. 220, n. 4598, p. 671-680.

[153] C. Papademetriou & K. Steiglitz. "Combinatorial optimization". 1982. *Englewood Cli s, NJ : Prentice Hall*.

[154] F. Glover. "Future paths for integer programming and links to artificial intelligence". *Computers and operations research*, 1986, vol. 13, n. 5, p. 533-549.

[155] T. A. Feo & M. G.C. Resende. "Greedy randomized adaptive search procedures". *Journal of global optimization*, 1995, vol. 6, n. 2, p. 109-133.

[156] N. Mladenovic & P. Hansen. "Variable neighborhood search". *Computers and operations research*, 1997, vol. 24, n. 11, p. 1097-1100.

[157] T. Stutzle. "Local search algorithms for combinatorial problems". *Thèse doctorat, l'université Darmstadt of Technology*, vol. 20, 1998.

[158] H. Salimi. "Stochastic fractal search : a powerful metaheuristic algorithm". *Knowledge-Based Systems*, 2015, vol. 75, p. 1-18.

[159] S. Mirjalili. "Dragonfly algorithm : a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems". *Neural Computing and Applications*, 2016, vol. 27, n. 4, p. 1053-1073.

[160] S. J. Mousavirad & H. Ebrahimpour-Komleh. “Human mental search : a new population-based metaheuristic optimization algorithm”. *Applied Intelligence*, 2017, vol. 47, n. 3, p. 850-887.

[161] S. Mirjalili. “SCA : a sine cosine algorithm for solving optimization problems”. *Knowledge-Based Systems*, 2016, vol. 96, p. 120-133.

[162] M. Douiri, S. Elbernoussi & H. Lakhba. “Cours des méthodes de résolution exactes heuristiques et métaheuristiques”. *Cours, l’université Mohamed V, Faculté des sciences de Rabat*, 2009, p. 5-87.

[163] “https ://fr.wikipedia.org/wiki/Métaheuristique”. *site web*. visité le 14/12/2018.

[164] M. Widmer. “Les métaheuristiques : des outils performants pour les problèmes industriels”. *3ème Conférence Francophone de MOdélisation et SIMulation MOSIM*. 2001, vol. 1, p. 25-27.

[165] J. H. Holland & J. S. Reitman. “Cognitive systems based on adaptive algorithms”. Pattern-directed inference systems, 1978, p. 313-329. *Elsevier*.

[166] I. Rechenberg. “Evolutions strategie, Optimierung technisher Systeme nach Prinzipien der biologischen Evolution”. *journal*, 1973.

[167] L. J. Fogel, A. J. Owens & M. J. Walsh. “Artificial intelligence through simulated evolution”. *journal*, 1966.

[168] M. Mitchell. “An introduction to genetic algorithms”. 1998. *MIT press*.

[169] R. Storn & K. Price. “Differential evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces [R]”. *Berkeley : ICSI*, 1995.

[170] M. Crepinsek, S. H. Liu & M. Mernik. “Exploration and exploitation in evolutionary algorithms : A survey”. *ACM Computing Surveys (CSUR)*, 2013, vol. 45, n. 3, p. 35.

[171] I. Boussaid. “Perfectionnement de métaheuristiques pour l’optimisation continue”. *Thèse doctorat : Informatique, l’université Paris Est*, 2013.

[172] R. Eberhart & J. Kennedy. “A new optimizer using particle swarm theory”. *MHS’95, Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. IEEE’1995, p. 39-43.

[173] K. M. Passino. “Biomimicry of bacterial foraging for distributed optimization and control”. *IEEE control systems*, 2002, vol. 22, n. 3, p. 52-67.

[174] Y. Shiqin, J. Jianjun & Y. Guangxing. “A dolphin partner optimization”. *Global Congress on Intelligent Systems*. IEEE’2009, p. 124-128.

[175] M. Dorigo, V. Maniezzo & A. Colorni. “Positive feedback as a search strategy”. *citeseerx*, 1991.

[176] S. Mirjalili, S. M. Mirjalili & A. Lewis. “Grey wolf optimizer”. *Advances in Engineering Software*, 2014, vol. 69, p. 46-61.

[177] A. Hatamlou. “Black hole : A new heuristic optimization approach for data clustering”. *Information Sciences*, 2013, vol. 222, p. 175-184.

[178] E. Rashedi, H. Nezamabadi-Pour & S. Saryazdi. “GSA : a gravitational search algorithm”. *Information Sciences*, 2009, vol. 179, n. 13, p. 2232-2248.

[179] O. K. Erol & I. Eksin. “A new optimization method : big bang-big crunch”. *Advances in Engineering Software*, 2006, vol. 37, n. 2, p. 106-111.

[180] H. Shah-Hosseini. “Principal components analysis by the galaxy-based search algorithm : a novel metaheuristic for continuous optimisation”. *International Journal of Computational Science and Engineering*, 2011, vol. 6, n. 1-2, p. 132-140.

[181] A. H. Kashan. “A new metaheuristic for optimization : optics inspired optimization (OIO)”. *Computers & Operations Research*, 2015, vol. 55, p. 99-125.

[182] A. Kaveh & V. R. Mahdavi. “Colliding bodies optimization : a novel meta-heuristic method”. *Computers and Structures*, 2014, vol. 139, p. 18-27.

[183] A. H. Kashan. “League Championship Algorithm (LCA) : An algorithm for global optimization inspired by sport championships”. *Applied Soft Computing*, 2014, vol. 16, p. 171-200.

[184] R. V. Rao, V. J. Savsani & DP. Vakharia. “Teaching-learning-based optimization : a novel method for constrained mechanical design optimization problems”. *Computer-Aided Design*, 2011, vol. 43, n. 3, p. 303-315.

[185] A. Sadollah, A. Bahreininejad, H. Eskandar, *et al.* “Mine blast algorithm : A new population based algorithm for solving constrained engineering optimization problems”. *Applied Soft Computing*, 2013, vol. 13, n. 5, p. 2592-2612.

[186] Z. Ibrahim, N. H. A. Aziz, N. A. Ab Aziz, *et al.* “Simulated Kalman filter : a novel estimation-based metaheuristic optimization algorithm”. *Advanced Science Letters*, 2016, vol. 22, n. 10, p. 2941-2946.

[187] D. H. Wolpert & W. G. Macready. “No free lunch theorems for optimization”. *IEEE Transactions on Evolutionary computation*, 1997, vol. 1, n. 1, p. 67-82.

[188] G. Cohen. “Convexité et optimisation”. *Cours, Ecole Nationale des Ponts et Chaussées*, p. 139, 2000.

[189] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, *et al.* “Equation of state calculations by fast computing machines”. *The journal of chemical physics*, 1953, vol. 21, n. 6, p. 1087-1092.

[190] Z. W. Geem, J. H. Kim & G. V. Loganathan. “A new heuristic optimization algorithm : harmony search”. *simulation*, 2001, vol. 76, n. 2, p. 60-68.

[191] F.Z. Benayed, L. Abdelhakem-Koridak & M. Rahli. “Optimisation d’un dispatching economique de la production combinee d’electricite et de la chaleur par l’algorithme harmony search”. *Mediamira Science Publisher*, 2011, vol. 52.

[192] A. Liefooghe. “Métaheuristiques pour l’optimisation multiobjectif : Approches coopératives, prise en compte de l’incertitude et application en logistique”. *Thèse doctorat : Informatique, l’université des Sciences et Technologie de Lille-Lille I*, 2009.

[193] A. Toumi. “Restauration adaptative d’image par les méthodes intelligentes”. *Thèse doctorat : Électronique, l’université Biskra, Algérie*, 2013.

[194] N. A. Kallioras, N. A. Lagaros & D. N. Avtzis. “Pity beetle algorithm-A new metaheuristic inspired by the behavior of bark beetles”. *Advances in Engineering Software*, 2018, vol. 121, p. 147-166.

[195] M. A. Diaz-Cortés, E. Cuevas & J. Gálvez, *et al.* “A new metaheuristic optimization methodology based on fuzzy logic”. *Applied Soft Computing*, 2017, vol. 61, p. 549-569.

[196] J. Chen, H. Cai & W. Wang. “A new metaheuristic algorithm : car tracking optimization algorithm”. *Soft Computing*, 2018, vol. 22, n. 12, p. 3857-3878.

[197] X. Qi, Y. Zhu & H. Zhang. “A new meta-heuristic butterfly-inspired algorithm”. *Journal of Computational Science*, 2017, vol. 23, p. 226-239.

[198] S. A. Ahmadi. “Human behavior-based optimization : a novel metaheuristic approach to solve complex optimization problems”. *Neural Computing and Applications*, 2017, vol. 28, n. 1, p. 233-244.

[199] M. Abdel-Basset, D. El-Shahat & A. K. Sangaiah. “A modified nature inspired metaheuristic whale optimization algorithm for solving 0-1 knapsack problem”. *International Journal of Machine Learning and Cybernetics*, 2017, p. 1-20.

[200] I. Strumberger & N. Bacanin. “Modified Moth Search Algorithm for Global Optimization Problems”. *International Journal of Computers*, 2018, vol. 3.

[201] J. S. Chou & N. T. Ngo. “Modified firefly algorithm for multidimensional optimization in structural design problems”. *Structural and Multidisciplinary Optimization*, 2017, vol. 55, n. 6, p. 2013-2028.

[202] M. Maadi, M. Javidnia & R. Ramezani. “Modified Cuckoo Optimization Algorithm (MCOA) to solve Precedence Constrained Sequencing Problem (PCSP)”. *Applied Intelligence*, 2018, vol. 48, n. 6, p. 1407-1422.

[203] B. Alizadeh & S. Bakhteh. “A modified firefly algorithm for general inverse p-median location problems under different distance norms”. *Opsearch*, 2017, vol. 54, n. 3, p. 618-636.

[204] K. Chen, F. Zhou & A. Liu. “Chaotic dynamic weight particle swarm optimization for numerical function optimization”. *Knowledge-Based Systems*, 2018, vol. 139, p. 23-40.

[205] Y. C. Wu, W. P. Lee & C. W. Chien. “Modified the performance of differential evolution algorithm with dual evolution strateg”. *International conference on machine learning and computing*. 2011, vol. 3, p. 57-63.

[206] P. Loubiere. “Amélioration des métaheuristiques d’optimisation à l’aide de l’analyse de sensibilité”. *Thèse doctorat : Mathématiques, Sciences et Technologies de l’Information et de la Communication, l’université Paris Est*, 2016.

[207] A. Milad. “Harmony search algorithm : strengths and weaknesses”. *Journal of Computer Engineering and Information Technology*, 2013.

[208] S. Khalilpourazari & S. Khalilpourazary. “A Robust Stochastic Fractal Search approach for optimization of the surface grinding process”. *Swarm and Evolutionary Computation*, 2018, vol. 38, p. 173-186.

[209] M. A. Mellal & E. Zio. “A penalty guided stochastic fractal search approach for system reliability optimization”. *Reliability Engineering & System Safety*, 2016, vol. 152, p. 213-227.

[210] J. Lin & Z. J. Wang. “Multi-area economic dispatch using an improved stochastic fractal search algorithm”. *Energy*, 2019, vol. 166, p. 47-58.

[211] S. Hinojosa, K. G. Dhal & M. A. Elaziz, *et al.* “Entropy-based imagery segmentation

for breast histology using the Stochastic Fractal Search". *Neurocomputing*, 2018, vol. 321, p. 201-215.

[212] X. Li, N. Xiao & C. Claramunt, *et al.* "Initialization strategies to enhancing the performance of genetic algorithms for the p-median problem". *Computers & Industrial Engineering*, 2011, vol. 61, p. 1024-1034, n. 4.

[213] N. Xiao. "A unified conceptual framework for geographical optimization using evolutionary algorithms". *Annals of the Association of American Geographers*, 2008, vol. 98, p. 795-817, n. 4.

[214] S. Deviant. "The practically cheating statistics handbook". 2011. *Lulu. com*.

[215] A. A. Goshtasby. "Image registration : Principles, tools and methods". 2012. *Springer Science and Business Media*.

[216] J. Lu & M. L. Liou. "A simple and efficient search algorithm for block-matching motion estimation". *IEEE Transactions on Circuits and Systems for Video Technology*, 1997, vol. 7, n. 2, p. 429-433.

[217] Z. Ahmed, A. J. Hussain & D. Al-Jumeily. "Mean Predictive Block Matching (MPBM) for fast block-matching motion estimation". *3rd European Workshop on Visual Information Processing*. IEEE'2011, p. 67-72.

[218] S. K. Sahu & D. Shukla. "A New Approach of Block Matching Motion Estimation Algorithm for H. 264/AVC Video Codec". *i-manager's Journal on Pattern Recognition*, 2017, vol. 4, n. 2, p. 10.

[219] G. G. Wang. "Moth search algorithm : a bio-inspired metaheuristic algorithm for global optimization problems". *Memetic Computing*, 2018, p. 1-14.

[220] B. Damerchilu, M. S. Norouzzadeh & M. R. Meybodi. "Motion estimation using learning automata". *Machine Vision and Applications*, 2016, vol. 27, n. 7, p. 1047-1061.

[221] T. Ojala, M. Pietikainen & D. Harwood. "Performance evaluation of texture measures

with classification based on Kullback discrimination of distributions". *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 1-Conference A : Computer Vision and Image Processing*. IEEE'1994, vol. 1, p. 582-585.

[222] T. Ojala, M. Pietikainen & D. Harwood. "A comparative study of texture measures with classification based on featured distributions". *Pattern recognition*, 1996, vol. 29, n. 1, p. 51-59.

[223] D. Huang, C. Shan, M. Ardabilian, *et al.* "Local binary patterns and its application to facial image analysis : a survey". *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2011, vol. 41, n. 6, p. 765-781.

[224] L. Nanni, A. Lumini & S. Brahnam. "Local binary patterns variants as texture descriptors for medical image analysis". *Artificial intelligence in medicine*, 2010, vol. 49, n. 2, p. 117-125.

[225] Z. Guo, L. Zhang & D. Zhang. "A completed modeling of local binary pattern operator for texture classification". *IEEE Transactions on image processing*, 2010, vol. 19, n. 6, p. 1657-1663.

[226] M. Pietikainen & G. Zhao. "Two decades of local binary patterns : A survey". Advances in independent component analysis and learning machines, 2015, p. 175-210. *Elsevier*.

[227] L. Liu, P. Fieguth, Y. Guo, *et al.* "Local binary features for texture classification : taxonomy and experimental study". *Pattern recognition*, 2017, vol. 62, p. 135-160.

[228] J. Ren, X. Jiang & J. Yuan. "Noise-resistant local binary pattern with an embedded error-correction mechanism". *IEEE Transactions on image processing*, 2013, vol. 22, n. 10, p. 4049-4060.

[229] L. Liu, L. Zhao, Y. Long, *et al.* "Extended local binary patterns for texture classification". *Image and Vision Computing*, 2012, vol. 30, n. 2, p. 86-99.

[230] D. T. Nguyen, P. O. Ogunbona & W. Li. "A novel shape-based non-redundant local

binary pattern descriptor for object detection". *Pattern recognition*, 2013, vol. 46, n. 5, p. 1485-1500.

[231] X. Qi, R. Xiao, C. G. Li, *et al.* "Pairwise rotation invariant co-occurrence local binary pattern". *IEEE transactions on pattern analysis and machine intelligence*, 2014, vol. 36, n. 11, p. 2199-2213.

[232] R. Maani, S. Kalra & Y. H. Yang. "Rotation invariant local frequency descriptors for texture classification". *IEEE Transactions on image processing*, 2013, vol. 22, n. 6, p. 2409-2419.

[233] J. Chen, S. Shan, C. He, *et al.* "WLD : A robust local image descriptor". *IEEE transactions on pattern analysis and machine intelligence*, 2010, vol. 32, n. 9, p. 1705-1720.

[234] J. Zhang, J. Liang & H. Zhao. "Local energy pattern for texture classification using self-adaptive quantization thresholds". *IEEE Transactions on image processing*, 2013, vol. 22, n. 1, p. 31-42.

[235] Z. Guo, X. Wang, J. Zhou, *et al.* "Robust texture image representation by scale selective local binary patterns". *IEEE Transactions on image processing*, 2016, vol. 25, n. 2, p. 687-699.

[236] M. Pietikainen. "Image analysis with local binary patterns". *Scandinavian Conference on Image Analysis*. Springer'2005, p. 115-118.

[237] M. Pietikainen, A. Hadid, G. Zhao, *et al.* "Computer vision using local binary patterns". Springer Science and Business Media, vol. 40. *2011*.

[238] J. E. Baker. "Reducing bias and inefficiency in the selection algorithm". *Proceedings of the second international conference on genetic algorithms*. 1987, vol. 206, p. 14-21.

[239] D. Karaboga. "An idea based on honey bee swarm for numerical optimization". *Technical report-tr06, Erciyes university, engineering faculty, computer engineering department*, 2005.

[240] R. G. Reynolds. “An introduction to cultural algorithms”. *Proceedings of the third annual conference on evolutionary programming*. World Scientific’1994, p. 131-139.

[241] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, *et al.* “Salp Swarm Algorithm : A bio-inspired optimizer for engineering design problems”. *Advances in Engineering Software*, 2017, vol. 114, p. 163-191.

[242] A. R. Hedar & A. F. Ali. “Tabu search with multi-level neighborhood structures for high dimensional problems”. *Applied Intelligence*, 2012, vol. 37, n. 2, p. 189-206.

[243] F. C. Zarrouk. “Les statistiques inférentielles, test de Student”. *Cours de statistiques à distance, ISSEP Ksar-Said*, 2012.

[244] N. Terki, D. Saigaa, L. Cheriet, *et al*. “Fast motion estimation algorithm based on complex wavelet transform”. *Journal of Signal Processing Systems*, 2013, vol. 72, no 2, p. 99-105.

[245] A. Toumi, N. Rechid, A. Taleb-Ahmed, *et al*. “Search efficiency function-PSO combination for blind image restoration”. *6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT). IEEE,* 2012, p. 293-296.

Annexe A

La fonction posynomiale

Une fonction $h(X)$ est posynomiale si h peut être exprimée comme une somme des termes, où chaque terme prend la forme suivante :
$c_i x_1^{ai1} x_2^{ai2} \cdots x_n^{ain}$
Où c_i et aij sont des constantes positives, donc un posynomial de N termes peut être exprimé comme suit :
$h(X) = c_1 x_1^{a11} x_2^{a12} \cdots x_n^{ain} + \cdots + c_N x_1^{aN1} x_2^{aN2} \ldots x_n^{aNn}$

La méthode de sélection roulette ou loterie biaisée

La méthode de sélection par roulette (Roulette Wheel Selection ou RWS) a été introduite par *Goldberg*, dont laquelle la population est représentée comme une roue de roulette, où chaque individu est représenté par une portion qui correspond proportionnellement à sa valeur de fitness. La sélection d'un individu se fait en tournant la roue en face d'un pointeur fixe. Quand elle cesse de tourner on séléctionne l'individu correspondant au secteur désigné par le pointeur. Avec une telle sélection, un individu fort peut être choisi plusieurs fois. Par contre, un individu faible aura moins de chance d'être sélectionné.

Annexe B

Fonctions de test	Dimensions	Intervalle	Nombre des itérations
$F_{01} = \sum_{i=1}^{n} x_i^2$	30	[-100,+100]	500
$F_{02} = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,+10]	950
$F_{03} = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	[-100,+100]	500
$F_{04} = \max_i \{ \lvert x_i \rvert, 1 \leq i \leq D \}$	30	[-100,+100]	1000
$F_{05} = \sum_{i=1}^{D-1} [100 \times (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,+30]	8000
$F_{06} = \sum_{i=1}^{D} ([x_i + 0.5])^2$	30	[-100,+100]	15
$F_{07} = \sum_{i=1}^{D} [i x_i^4 + random[0,1)]$	30	[-1.28,+1.28]	1500
$F_{08} = \sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500,+500]	1500
$F_{09} = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,+5.12]	40
$F_{10} = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})$ $-exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	30	[-32,+32]	60
$F_{11} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600,+600]	70
$F_{12} = \frac{\pi}{D}\{10\sin^2(\pi y_i)$ $+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_i+1)]$ $+(yD-1)^2 + \sum_{i=1}^{D} u(x_i,10,100,4)\}$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i-a)^m & x_i < -a \end{cases}$	30	[-50,+50]	2000
$F_{13} = 0.1\{10\sin^2(\pi y_i)$ $+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_i+1)]$ $+(yD-1)^2 + \sum_{i=1}^{D} u(x_i,10,100,4)\}$	30	[-50,+50]	2000
$F_{14} = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}]^{-1}$	2	[-65.53,+65.53]	150
$F_{15} = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_i)}{b_i^2 + b_1 x_3 + x_4}]^2$	4	[-5,+5]	400
$F_{16} = 4x_1^2 - 2.1x_i^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,+5]	200
$F_{17} = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	2	$[5,+10] \times [0,+15]$	180
$F_{18} = [1 + (x_1 + x_2 + 1)^2$ $(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times [30 + (2x_1 - 3x_2)^2$ $(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-5,+5]	200
$F_{19} = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	[0,+1]	100
$F_{20} = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	[0,+1]	250
$F_{21} = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,+10]	200
$F_{22} = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,+10]	200
$F_{23} = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,+10]	200

Annexe C

Extrait de la table de Student

Degré de liberté	Le seuil de probabilité					
	0.2	0.1	0.05	0.025	0.01	0.005
1	1.376	3.078	6.314	12.706	31.821	63.656
2	1.061	1.886	2.920	4.303	6.965	9.925
3	0.978	1.638	2.353	3.182	4.541	5.841
4	0.941	1.533	2.132	2.776	3.747	4.604
5	0.620	1.476	2.015	2.571	3.365	4.032
6	0.906	1.440	1.943	2.447	3.143	3.707
7	0.896	1.415	1.895	2.365	2.998	3.499
8	0.889	1.397	1.860	2.306	2.896	3.355
9	0.883	1.383	1.833	2.262	2.821	3.250
10	0.879	1.372	1.812	2.228	2.764	3.169
11	0.876	1.363	1.796	2.201	2.718	3.106
12	0.873	1.356	1.782	2.179	2.681	3.055
13	0.870	1.350	1.771	2.160	2.650	3.012
14	0.868	1.345	1.761	2.145	2.624	2.977
15	0.866	1.341	1.753	2.131	2.602	2.947
16	0.865	1.337	1.746	2.120	2.583	2.921
17	0.863	1.333	1.740	2.110	2.567	2.898
18	0.862	1.330	1.734	2.101	2.552	2.878
19	0.861	1.328	1.729	2.093	2.539	2.861
20	0.860	1.325	1.725	2.086	2.528	2.845
:	:	:	:	:	:	:
48	0.849	1.299	1.677	2.011	2.407	2.682

Annexe D

Les sources de code utilisées dans cette thèse ont été téléchargées à partir des adresses suivantes :

- CA :`http://yarpiz.com/`
- SCA : `https://www.mathworks.com/matlabcentral/fileexchange/54948-sca--a-sine-cosine-algorithm`
- GWO : `https://www.mathworks.com/matlabcentral/fileexchange/47258-grey-wolf-optimizer-toolbox`
- SCA : `https://www.mathworks.com/matlabcentral/fileexchange/54948-sca--a-sine-cosine-algorithm`
- SSA : `https://www.mathworks.com/matlabcentral/fileexchange/63745-ssa-salp-swarm-algorithm`
- MS : `https://www.mathworks.com/matlabcentral/fileexchange/59010-moth-search-ms-algorithm`
- BM : `https://www.mathworks.com/matlabcentral/fileexchange/8761-block-matching-algorithms-for-motion-estimation`

Les séquences vidéo utilisées pour estimation de mouvement sont téléchargées à partir de ces adresses :

- `http://trace.kom.aau.dk/yuv/index.html`
- `https://media.xiph.org/video/derf/`