République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Sadik Benyahia de Jijel
Faculté des Sciences Exactes et informatique
Département d'Informatique

Mémoire de fin d'étude

pour obtention du diplôme Master de Recherche en Informatique

Option : *Informatique légal et multimédia*

Thème

Parrallélisation en îlots avec migration d'un cryptage génétique d'une image

Présenté par :
Hadji Asma .
Bounesrag Amal.

Encadré par :
Dr.Ismahane Souici

Promotion : 2019.

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Sadik Benyahia de Jijel
Faculté des Sciences Exactes et informatique
Département d'Informatique

Mémoire de fin d'étude

pour obtention du diplôme Master de Recherche en Informatique

Option : *Informatique légal et multimédia*

Thème

Parrallélisation en îlots avec migration d'un cryptage génétique d'une image

Présenté par :
Hadji Asma .
Bounesrag Amal.

Encadré par :
Dr.Ismahane Souici

Promotion : 2019.

⁎ REMERCIEMENTS ⁎

Avant tout, nous tenons à remercier Allah le tout-puissant et miséricordieux de nous avoir donnée assez de courage, de volonté et de force pour mener à terme ce projet de fin d'études.

Nous tenons à remercier notre encadreur ¡¡ Mme Souici Ismahane ¿¿ pour sa patience et sa compréhension, pour sa présence chaque fois que nous en avons besoin, pour nous avoir guidé, encouragé et aidé à mener à bien ce travail. Notre sincère gratitude va pour elle.

à nos familles qui nous ont soutenus, encouragés et aidés à devenir les personnes qu'elles sont aujourd'hui.

Nous remercions également, tous les enseignants qui ont assurés notre formation durant notre cycle universitaire de Jijel.

Nous remercions les membres de jury qui ont accepté de jurer notre travail. Qu'ils trouvent ici, l'expression de notre profond, respect et notre sincère gratitude. Merci à tous ceux qui nous ont aidés à terminer notre travail. A tous, un grand merci

⁂ Dédicaces ⁂

A tous ceux qui étaient là pour moi dans les meilleurs et les pires moments de ma vie, qui ont soutenu, encouragé, aidé à devenir la personne que je suis aujourd'hui.

A mes parents, ma petite soeur Manel. ,

Aux personnes les plus proches de moi, qui me connait vraiment et m'accepte comme je suis sans conditions.

A vous tous, je vous aime et je dédie ce travail pour vous,

Asma

⁎ Dédicaces ⁎

A mes Parents,
A mes frères ,

S es soutiens moral et leurs conseils précieux tout au long de mes études.

Sans oublier bien sûr mon binôme Asma.
A mes collègues Master 2 Informatique légale et multimédia .

A tous qui ont contribué de prés ou de loin à la réalisation de ce travail. **Amel**

Résumé

Le domaine de la cryptographie est en constante évolution car la protection des informations personnelles a toujours été l'objectif de l'homme. Pour cette raison, nous avons choisi de participer à ce développement en enrichissant le domaine du cryptage génétique grâce à une nouvelle méthode offrant une solution de qualité au problème de cryptage des images.
Bien que les algorithmes génétiques puissent être considérés comme une solution parfaite au problème de cryptage d'image, car le résultat de chacune de ses étapes est produit de manière aléatoire et ne peut être prédit ; Cependant, les algorithmes génétiques séquentiels ne peuvent pas être utilisés dans le cas de populations de grande taille, car ils consomment beaucoup de temps et de ressources. Pour cette raison, nous avons choisi de tirer parti de l'algorithme génétique parallèle et plus précisément du modèle d'îlots avec migration car il exécute toutes les étapes d'un algorithme génétique séquentiel en mode parallèle.
PosESecL2 est une méthode de cryptage génétique, développée à l'aide d'un algorithme génétique séquentiel. C'est également l'algorithme de base sur lequel les deux versions parallèles PFSPos et PVSPos ont été construites afin de réduire le temps de calcul en utilisant le modèle d'îlots sans migration.
Dans notre travail, nous avons développé un algorithme basé sur PFSPos en rajoutant une étape de la migration. La mise en oeuvre a été effectuée à l'aide d'un modèle d'îlot avec migrations. Le résultat était l'algorithme PFSPosMig.
Nous avons effectué une série de tests sur l'algorithme PFSPosMig afin de fixer les paramètres relatifs à l'opération de migration. De même, une comparaison entre ces trois alternatives de cryptage génétiques parallèles, basée sur des tests expérimentaux a été réalisé.

Abstract

The field of cryptography is constantly changing because the protection of personal information has always been the goal of man. For this reason, we chose to participate in this development by enriching the field of genetic encryption with a new method offering a quality solution to the problem of image encryption.
Although genetic algorithms can be considered a perfect solution to the problem of image encryption, because the result of each of its steps is produced randomly and cannot be predicted ; However, sequential genetic algorithms cannot be used in large populations because they consume a lot of time and resources. For this reason, we chose to take advantage of the parallel genetic algorithm and more precisely the migration island model because it performs all the steps of a sequential genetic algorithm in parallel mode. PosESecL2 is a method of genetic encryption, developed using a sequential genetic algorithm. It is also the basic algorithm on which the two parallel versions PFSPos and PVSPos were built to reduce the computation time using the island model without migration.
In our work, we developed an algorithm based on PFSPos by adding a step of the migration. The implementation was performed using an island model with migrations. The result was the PFSPosMig algorithm.
We performed a series of tests on the PFSPosMig algorithm to set the parameters for the migration operation. Likewise, a comparison between these three alternative genetic encryption alternatives based on experimental tests has been made.

Table des matières

Liste des tableaux

Table des figures

Liste des algorithmes

Liste des abréviations

RVB : Rouge Vert Bleu

Introduction générale

Protéger les données sensibles des yeux des gens curieux ou des ennemis a été l'objectif de nombreuses civilisations au cours des siècles passés. Au XXIe siècle, la sécurisation des données est plus qu'importante ; c'est une chose nécessaire à faire en raison de l'énorme quantité de données existantes dans le monde numérique et qui peut être une cible pour les pirates et les vendeurs de données qui ne se soucient pas de la vie privée des personnes.
L'évolution des technologies, en particulier en informatique, a rendu de nombreuses institutions officielles dépendantes d'elles pour exécuter différents services. Cela inclut la sauvegarde de données sensibles pouvant être des dossiers médicaux, des informations sur des comptes bancaires, etc. De plus, les données sensibles n'existent pas seulement dans les institutions officielles, mais également dans les serveurs de réseaux sociaux (photos, informations privées, messages privés, mots de passe, etc.), les sites de shopping (informations de paiement, détails relatifs au crédit cartes, etc.) également, l'ordinateur d'un utilisateur ordinaire peut contenir des données sensibles telles que des fichiers de travail, des mots de passe pour différents sites et des e-mails...
Pour cette raison, il est important d'utiliser des outils et des méthodes offrant un niveau de sécurité maximal et évitant que des données sensibles ne tombent entre les mains de personnes curieuses et des mauvais. Une de ces méthodes est la cryptographie.
La cryptographie est l'une des méthodes permettant de garantir la sécurité des données de différentes manières ; si ces données sont enregistrées ou transférées. Elle offre de nombreux avantages lors de son utilisation : les mots de passe des utilisateurs peuvent être conservés et protégés, les informations financières peuvent être transférées et sauvegardées dans des bases de données en toute sécurité et les communications en ligne entre différentes personnes peuvent être gardées privées et protégées.
Afin de fournir cette sécurité aux différents types de données, l'application de méthodes de sécurité peut rencontrer des problèmes qui ne peuvent pas être résolus en utilisant des algorithmes déterministes. Pour cette raison, les algorithmes génétiques peuvent être la solution à ces problèmes ; le choix du type d'algorithme génétique à utiliser est défini par le type d'application et les résultats attendus.
Dans le premier chapitre, le concept de cryptage des données et ses types seront introduits, tandis que dans le deuxième chapitre, la définition de l'algorithme génétique et son fonctionnement seront vus avec leurs types et modèles.

Dans le troisième chapitre, une description des méthodes et des algorithmes utilisés pour produire l'algorithme principal qui sera implémenté ultérieurement sera présentée. Dans le dernier chapitre, la structure de l'application sera décrite, ainsi que les expériences qui ont été effectuées et leurs résultats.

Chapitre 1

Cryptage de données

1.1 Introduction

Pendant des siècles, les humains ont essayé de cacher leurs informations à d'autres personnes spécialement leurs ennemis. Pour cette raison , ils ont créé différentes méthodes et les ont utilisées dans de nombreux domaines où ils ont gardé leurs secrets en les transformant en codes secrets. Par exemple , il y a environ 4000 ans, les Égyptiens utilisaient des hiéroglyphes pour écrire des messages. Le code était secret et n'était connu que des scribes qui transmettaient des messages au nom des rois.Durant la période allant de 500 à 600 av. J.-C., les érudits ont utilisé une méthode simple de chiffrement de substitution mono-alphabétique : ils ont remplacé les alphabets du message par d'autres règles secrètes, et cette règle est devenue la clé à utiliser pour récupérer le message à partir du message tronqué. Également, les Hébreux dissimulaient parfois leurs écrits en inversant l'alphabet, c'est-à-dire en employant la dernière lettre de l'alphabet à la place de la première, l'avant dernière lettre à la place de la deuxième, et ainsi de suite.
En 100 BCE, Jules César [1] se servit également de codes secrets pour correspondre avec ses hommes, et laissa même son nom à un chiffre particulierqui repose sur le décalage des lettres d'un message par les lettres qui étaient les troisièmes après les lettres originales du message sur l'alphabet.
En 1467,chiffrement d'Alberti [2] fut inventé par LeonBattista Alberti et considéré comme le premier chiffre de substitution poly-alphabétique. Il était composé de deux disques métalliques sur le même essieu, l'un à l'intérieur de l'autre, qui comprenaient des alphabets mixtes et des rotations variables.Un autre exemple, la roue Jefferson [2], un outil de chiffrement inventé par Thomas Jefferson, alors qu'il était secrétaire d'État de George Washington en 1797.La roue était composée de 26 pièces de bois cylindriques enfilées sur une broche en fer. Les lettres de l'alphabet étaient inscrites sur le bord de chaque roue dans un ordre aléatoire. En le tournant, on obtenait le message chiffré et le message clair. Le destinataire épèlera le message codé sur sa roue, puis chercherait la ligne de lettres qui avait du sens.
Mais le développement des méthodes de cryptographie ne s'arrête pas là, il continue de croître de manière croissante, particulièrement dans la première et la seconde guerre, où les

parties aux conflits sont obsédées par la nécessité de sécuriser leurs lignes de communication afin d'empêcher les espions d'y accéder à des informations sensibles.
Au XXIe siècle, la cryptographie est utilisée dans presque tous les domaines, en particulier sur Internet, où la protection des informations personnelles est l'objectif de tous ceux qui proposent des services aux utilisateurs ou aux clients.
Dans ce chapitre, on apprend ce qu'est la cryptographie, quels sont ses principes et ses classes avec des exemples.

1.2 Terminologie

Le dictionnaire **Merriam Webster**[3] définit le mot cryptographie comme suit :

- ≪*secret writing* ≫ ce qui signifie : écriture secrète.
- ≪*the enciphering and deciphering of messages in secret code or cipheralso : the computerizedencoding and decoding of information* ≫ ce que signifie : le chiffrement et le déchiffrement des messages en code secret ou chiffré également : le codage et le décodage informatisés de l'information.

Le dictionnaire **La Rousse**[4] définit le mot cryptographie comme suit : ≪ *Ensemble des techniques de chiffrement qui assurent l'inviolabilité de textes et, en informatique, de données* ≫.
Le mot cryptographie qui vient de la combinaison des deux mots grec kruptos(caché)et graphein (écrire) est la discipline incluant les principes, les moyens et les méthodes de transformation des données, dans le but de masquer leur contenu, empêcher leur modification ou leur utilisation illégale, ainsi que les opérations inverses, pour rendre le document à nouveau intelligible.
Comme tout système, la cryptographie a ses propres mots.Par exemple, le message qui sera crypté afin de protéger les informations est appelé ≪texte en clair ≫, tandis que le résultat du processus de cryptage est appelé ≪texte crypté ≫, et l'outil de cryptage est la clé de cryptage.
Il existe deux types de clé de cryptage : la clé symétrique et la clé asymétrique. Chacune d'entre elles possède sa propre définition, comme on le verra dans les sections suivantes.
La cryptographie est la première partie de la cryptologie tandis que la seconde partie est la cryptanalyse.
Cryptologie : Il s'agit d'une science mathématique comportant deux branches : la cryptographie et la cryptanalyse. [5]
Cryptosystème : Il est défini comme l'ensemble des clés possibles (espace de clés), des textes clairs et chiffrés possibles associés à un algorithme donné.[5]
Chiffrement : Le chiffrement consiste à transformer une donnée (texte, message, ...) afin

de la rendre incompréhensible par une personne autre que celui qui a créé le message et celui qui en est le destinataire. La fonction permettant de retrouver le texte clair à partir du texte chiffré porte le nom de déchiffrement. [5]
Cryptanalyse : Opposée à la cryptographie, elle a pour but de retrouver le texte clair à partir de textes chiffrés en déterminant les failles des algorithmes utilisés.[5]

1.3 Principes et fonctionnement de la cryptographie

Le système cryptographique est principalement utilisé pour transformer des messages entre deux côtés ou de multiples côtés. Pour cette raison, la cryptographie a pour objectif de garder le secret des informations en préservant plusieurs fonctions.
Les plus importantes sont :

- ✓ **la confidentialité :** Qui signifie la protection des informations secrètes vis-à-vis des personnes non autorisées à le connaître.Cela signifie que si A a envoyé un message chiffré à B, personne à l'exception de B ne peut savoir quel est le message en clair.

- ✓ **L'intégrité :** Est la capacité de savoir que les informations d'origine ont été modifiées,Par conséquent, si A a envoyé un message à B, B peut savoir avec certitude que le message d'origine a été modifié.

- ✓ **L'authentification :** Est la confirmation de l'identité de l'expéditeur et du destinataire, si A a envoyé un message à B, A est sûr que B est celui qui recevra le message et B est sûr que A est celui qui a envoyé le message.

- ✓ **La non répudiation :** Consiste à garantir qu'aucune des parties ne peut nier la transaction ; cela signifie que A ne peut pas nier qu'il a envoyé le message et B ne peut pas nier qu'il a reçu le message.
  Le système cryptographique a la structure d'un algorithme où l'entrée est le texte en clair, le résultat est le texte crypté pour l'opération de cryptage, et dans le cas de déchiffrement le texte chiffré correspond à l'entrée et le texte en clair correspond au résultat. Le processus de chiffrement utilisant la clé de chiffrement constitue le corps de l'algorithme.

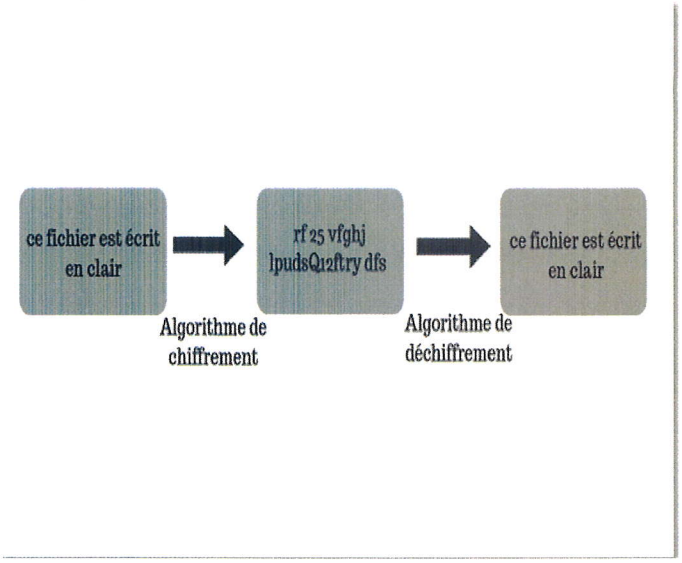


FIGURE 1.1 – le processus de la cryptographie.

Avant le XVIIII siècle, les utilisateurs du système cryptographique essayaient toujours de garder la méthode de cryptageutilisée secrète jusqu'à ce qu'Auguste Kerckhoff vienne avec l'idée de garder la clé de cryptage secrète et non l'algorithme lui-même afin de protéger les informations d'origine.Pour cette raison, il identifie six règles qui peuvent être suivies :

1. « *le système doit être matériellement, sinon mathématiquement, indéchiffrable.*
2. *Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénients tomber entre les mains de l'ennemi.*
3. *La clé doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants.*
4. *Il faut qu'il soit applicable à la correspondance télégraphique.*
5. *Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes.*
6. *Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.* » [6]

Outre le principe de kerckhoff, il existe deux autres principes. Le premier indique que le déchiffrement d'un texte chiffré sans la clé de déchiffrement est impossible [7], ce qui signifie qu'un intrus n'a pas la possibilité d'obtenir le texte en clair sans la clé. Le second indique qu'il est également impossible de trouver la clé de chiffrement à partir de texte chiffré et de texte en clair [7], par conséquent, un intrus peut avoir à la fois le texte chiffré et le texte en clair, mais il lui est impossible d'obtenir la clé de chiffrement.

La structure de l'algorithme de chiffrement peut être représentée par une équation mathématique :

$$c = e_k(m)$$

Où :
m : est le texte en clair.
e : est la fonction de chiffrement.
k : est la clé.
c : est le texte chiffré.

Et pour l'algorithme de déchiffrement, l'équation mathématique est la suivante :

$$m = d_k(c)$$

Où :
d : est la fonction de déchiffrement.
Les deux fonctions de chiffrement et de déchiffrement e,d sont publiques,
le secret de m étant c dépend totalement du secret de k [8]

1.4 Classes de la cryptographie

Dans cette section, nous résumons les différentes classes de cryptographie et les différents types de chacune d'elles et nous citons des exemples pour chaque type. La cryptographie regroupe deux classes : la cryptographie classique et la cryptographie moderne.

1.4.1 La cryptographie classique :

La cryptographie classique est représentée avec les méthodes cryptographiques utilisées au cours des siècles avant l'apparition de l'ordinateur. Le principe de ces méthodes était la manipulation des caractères du message original. cela se fait en faisant une substitution avec d'autres caractères en suivant des règles qui diffèrent d'une méthode à l'autre ; ou en utilisant une transposition qui consiste à manipuler l'ordre des caractères de différentes manières. Ainsi, on distingue deux types de cryptographie classique :

1.4.1.1 Avec substitution :

Un cryptage avec substitution consiste à remplacer les caractères ou les symboles par d'autres Deux types de substitution se distinguent :

✓ **Substitution mono-Alphabétique :** cette méthode remplace chaque lettre par une autre lettre ou symbole. Cela peut être formaliser comme suit :

$$f : A_M \rightarrow A_C$$

$$c_i = f(m_i) = m_i + k(mod\,|A_M|)$$

Où :

$$A_M, A_C$$

sont respectivement l'ensemble d'alphabets du message en clair et l'ensemble d'alphabets du message chiffré

$$M = m_0 m_1 \cdots m_{n-1}$$

telque :

$$\forall : i, m_i \in A_M$$

$$C = c_0 c_1 \cdots c_{n-1}$$

telque :

$$\forall : i, c_i \in A_C$$

Exemples : chiffrement de César [5], chiffrement Affine [5].

✓ **Substitution poly alphabétique :** elle a été inventée par Trithemius en 1518. La méthode consiste à remplacer une même lettre par plusieurs symboles, pris aléatoirement.Cela est garantie grâce à une clé k = k0 k1,,,, kj-1 qui définit j fonctions distinctes fk1, fk2,,,, fkj-1 définies comme suit :

$$\forall i : 0 \leq i < n$$

$$f_{k1} : A_M \rightarrow A_C$$

$$\forall i : 0 \leq l < j$$

$$c_i = f_{k_{imodj}}(m_i) = m_i + k_{imodj}(mod \mid A_M \mid$$

Avec AM, AC, M et C auront la même signification que ceux utilisés dans la définition de la substitution mono-alphabétique.

Exemple : Le chiffrement de Vigenère[9]

1.4.1.2 Par transposition :

Transposer signifie manipuler l'ordre des caractères et non les caractères eux-mêmes. Plusieurs types de chiffrement par transposition se distinguent, parmi lesquels nous citons les suivants :

✓ **Transposition simple par colonnes :** Le message en clair est écrit ligne par ligne dans une matrice prédéfinie et le texte chiffré est trouvé en lisant la grille colonne par colonne. Le processus inverse est effectué afin de déchiffrer le texte chiffré.
La figure 1.2 présente un exemple de transposition simple par colonnes.

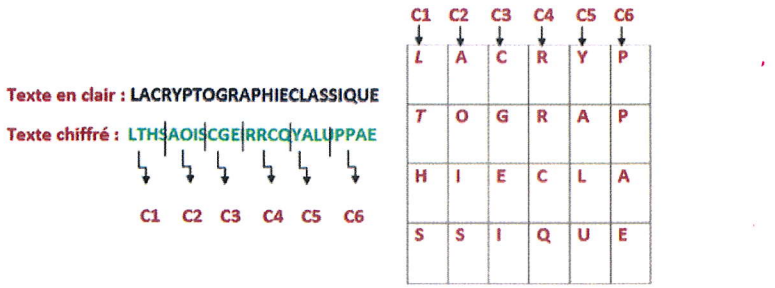


FIGURE 1.2 – Transposition simple par colonne

✓ **Transposition complexe par colonnes :** Dans ce type de chiffrement, une clé secrète est utilisée pour obtenir le texte chiffré à partir du message en clair. Cela se fait en donnant à chaque lettre composant la clé secrète un numéro commençant de gauche à droite et en suivant l'ordre d'apparition de chaque lettre dans l'alphabet. Une fois que la séquence des nombres est obtenue, le message en clair sera écrit dans un rectangle ligne par ligne et le texte chiffré sera obtenu en lisant ce que le rectangle contient colonne par colonne dans l'ordre déterminé par la séquence de nombres.
La figure 1.3 résume un exemple de transposition complexe par colonnes.

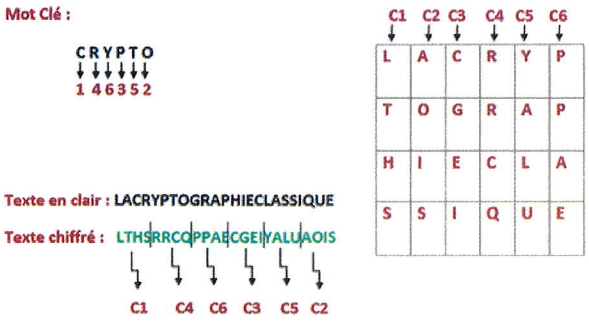


FIGURE 1.3 – Transposition complexe par colonne

✓ **Transposition par carré polybique :** Dans un tableau, un alphabet est construit à l'aide d'uneclé secrète, en commençant par les lettres de la clé secrète et en continuant avec le reste des lettres de l'alphabet. Un numéro spécifique est attribué à chaque ligne et colonne, et le numéro de la ligne et de la colonne correspondant à la lettre la représentera verticalement. Après cette étape, toutes les lettres du message en clair sont représentées par un nombre de lignes et de colonnes ; le texte chiffré est obtenu en divisant la séquence de nombres deux à deux horizontalement pour obtenir le nombre de lignes et de colonnes de chaque lettre constituant le texte chiffré.
La figure 1.4 expose un exemple de transposition par carré polybique.

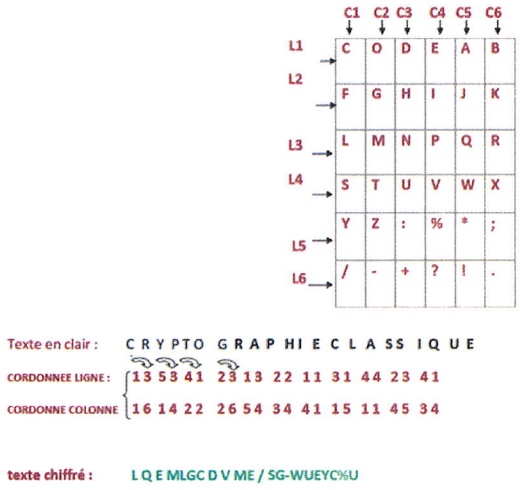


FIGURE 1.4 – transposition par carré polybique

1.4.2 La cryptographie moderne

La cryptographie moderne est apparue avec l'apparition des ordinateurs, qui repose sur divers concepts mathématiques et opère sur des séquences de bits.
La cryptographie moderne repose sur des algorithmes mathématiques connus du grand public. Le secret est ainsi obtenu grâce à une clé secrète qui est utilisée comme germe des algorithmes.

1.4.2.1 La cryptographie symétrique

Cette méthode de chiffrement dépend de l'utilisation d'une clé privée ou également appelée clé symétrique. Cette clé privée doit être connue des deux parties (émetteur et récepteur) et doit être transférée dans un canal sécurisé. La clé privée utilisée pour chiffrer le message est la même que celle utilisée pour déchiffrer le message.
Le principe de la cryptographie symétrique peut être résumé par les équations suivantes :
Cryptage symétrique :

$$A_C(k, m) = c$$

Décryptage symétrique :

$$A_C(k, c) = m$$

Où :
k : la clé privée.
m : le texte en clair.
c : le texte chiffrée.

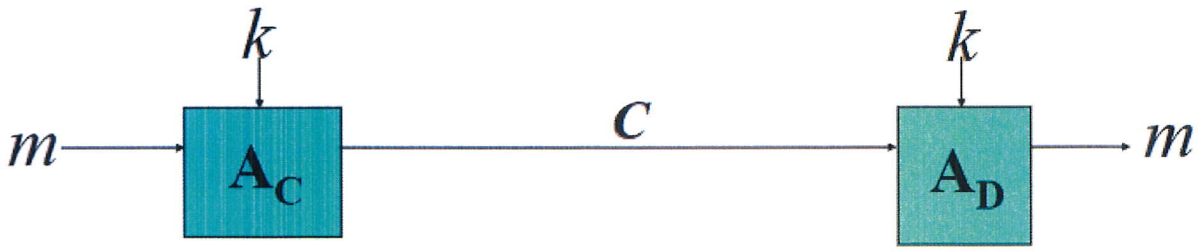


FIGURE 1.5 – la cryptographie symétrique [10]

Le processus de cryptage symétrique peut être effectué de deux manières :

- ✓ **Chiffrement par flots :** où le message sera traité comme des flots.
- ✓ **Chiffrement par bloc :** où le message sera divisé en blocs ayant la même taille.

Exemples : DES [11], AES [12], RC4 [13].

1.4.2.2 La cryptographie asymétrique :

La cryptographie asymétrique a été inventée par WhitfieldDiffie et Martin Hellman en 1976 [14].
Dans la cryptographie asymétrique, la clé utilisée pour chiffrer et déchiffrer le message est composée de deux parties : la première partie est la clé publique utilisée pour chiffrer le message et la seconde partie est la clé secrète utilisée pour déchiffrer le message chiffré.
La clé publique et la clé secrète sont choisies par le récepteur, et seule la clé publique est envoyée à l'émetteur dans un canal non sécurisé où il cryptera le message avec celle-ci. Le récepteur du message chiffré le déchiffrera à l'aide de la clé privée qu'il a et n'a pas été transférée.

Le principe de la cryptographie asymétrique peut être résumé comme suit :
Cryptage publique :

$$A_C(pk, m) = c$$

Décryptage publique :

$$A_C(sk, c) = m$$

Où :
pk : la clé publique.
sk : la clé privée.
m : le texte en clair.
c : le texte chiffré.
Exemple : RSA [15].

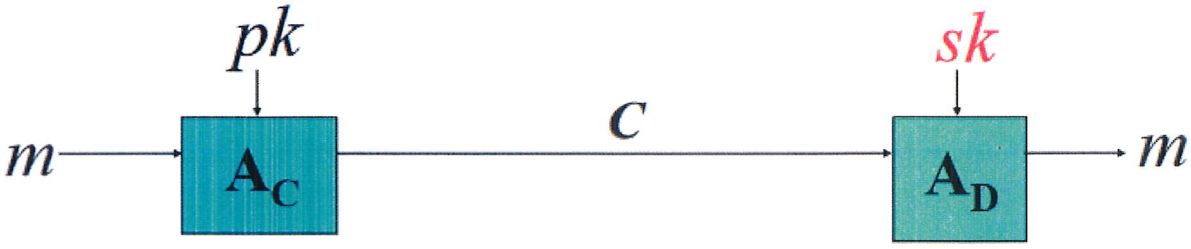


FIGURE 1.6 – la cryptographie asymétrique [10]

1.4.2.3 Les Avantages et Les inconvénients de la cryptographie symétrique et asymétrique

Les systèmes cryptographiques symétriques et asymétriques ont leurs avantages et leurs inconvénients, qui peuvent être résumés dans les deux tableaux suivants :

- *les Avantages*

Cryptographie symétrique	Cryptographie asymétrique
La taille de la clé est très courte. Le maximum est de 256 bits.	Pas besoin de transférer la clé privée car la clé publique est celle qui est utilisée pour le cryptage.
L'application de l'algorithme est facile à faire et très rapide.	Les paires de clés peuvent être utilisées pendant une longue période sans les changer.
La clé privée résiste de manière robuste aux attaques par force brute.	Fournit un échange sécurisé d'informations.

TABLE 1.1 – les avantages de la cryptographie symétrique et asymétrique

- *les Inconvénients*

La cryptographie symétrique	La cryptographie asymétrique
La nécessité d'échanger la clé privée est le principal problème.	L'algorithme est lent
La clé privée doit être changée constamment.	La taille de la clé est très longue, le maximum est de 4096 bits.

TABLE 1.2 – les inconvénients de la cryptographie symétrique et asymétrique

1.4.2.4 La cryptographie hybride :

En raison des problèmes des deux types qui ont été vus dans la sous-section précédente, les cryptographes ont construit un système cryptographique qui combine les deux types précédents dans un autre type afin de bénéficier des avantages des deux.
Le cryptosystème hybride est composé de cryptosystèmes symétriques et asymétriques.
Le cryptosystème symétrique est utilisé pour chiffrer le message en clair à l'aide d'une clé aléatoire, tandis que le cryptosystème asymétrique est utilisé pour chiffrer la clé aléatoire.
L'expéditeur génère une clé aléatoire qui sera utilisée pour chiffrer le message en clair. Après cela, il cryptera la clé aléatoire avec la clé publique du récepteur et le message de chiffrement avec la clé aléatoire cryptée sera envoyé. Le récepteur recevra la clé aléatoire cryptée avec le message de chiffrement et il décodera d'abord la clé aléatoire cryptée à l'aide de sa clé privée pour obtenir la clé aléatoire. Ensuite, il déchiffrera le message chiffré avec la clé aléatoire pour obtenir le message clair.

Exemple : PGP [16].

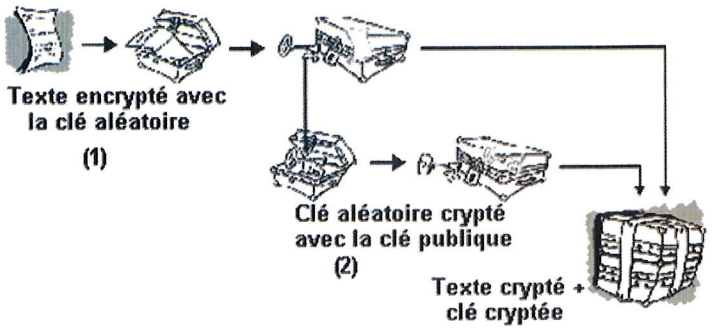


FIGURE 1.7 – Processus de chiffrement des données dans la cryptographie hybride.[17]

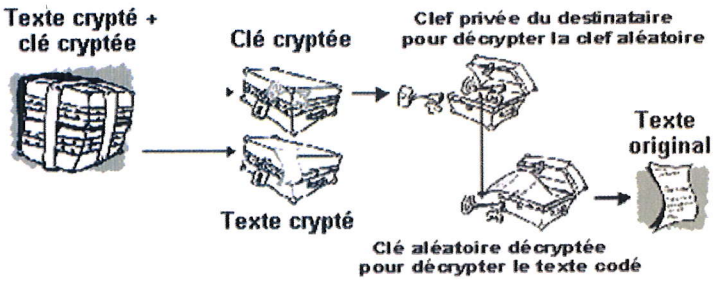


FIGURE 1.8 – Processus de déchiffrement des données dans la cryptographie hybride.[17]

Chapitre 2

Les algorithmes génétiques

2.1 Introduction

Les scientifiques et les chercheurs ont essayé au fil des siècles de trouver des solutions aux différents problèmes auxquels on peut faire face dans la vie ordinaire ou dans le domaine scientifique. Pour y parvenir, ils devaient parfois s'inspirer d'autres domaines scientifiques non liés à leur domaine afin de développer des méthodes capables de résoudre le problème ou d'introduire une meilleure solution à partir des multiples solutions proposées au problème. Les algorithmes génétiques font partie des méthodes introduites dans le domaine de l'informatique pour fournir une solution optimale parmi un groupe de solutions proposées pour résoudre des problèmes qui ne peuvent pas être résolus avec l'utilisation d'un algorithme déterministe. Ce chapitre présente ce qu'est un algorithme génétique, son fonctionnement, ses avantages et limites, ce qui est un algorithme génétique parallèle et leurs modèles.

2.2 Définition

Les algorithmes génétiques sont des algorithmes de recherche basés sur les mécanismes de la sélection naturelle et de la génétique naturelle [18].Ils sont stochastiques et font partie de la famille des algorithmes évolutionnaires inspirés du néo-darwinisme, de la théorie de la sélection naturelle de Charles Darwin et de la théorie de l'hérédité de Mendal, qui sont des théories proposées dans le domaine biologique. Leurs principes sont :
Evolution : résultat d'une altération progressive des êtres vivants au cours de générations ; Reproduction basée sur le caractère génétique qui subit au cours des générations des recombinaisons et des mutations ; Mécanisme de sélection naturelle[19] . Ils sont utilisés à des fins d'optimisation en obtenant la solution optimale à partir d'un groupe de solutions pour un problème spécifique.
La première apparition des algorithmes génétiques a été réalisée par Bremermann[20] en 1958, mais ils n'ont pas viré au virus jusqu'à ce que Holland commence à travailler sur les algorithmes génétiques. Il les a développés avec ses collègues et ses étudiants dans les années 1960. En 1968, le développement du schéma a été mis au point[21] et les détails du travail ont été publiés en 1975 [22].

Après cela, Goldberg qu'est un étudiant d' Holland, est venu et a rendu les algorithmes génétiques plus célèbres par ses travaux et ses recherches, qui ont été introduits dans son livre « genetic algorithms in search, optimization and machine learning » en 1989. Il a déclaré que les recherches effectuées par Holland, ses collègues et ses étudiants avaient deux objectifs : résumer et expliquer de façon régressive les processus d'adaptation des systèmes naturels et concevoir un logiciel de systèmes artificiels qui conserve le mécanisme important des systèmes naturels[18].

2.3 Terminologie

Les algorithmes génétiques ont leur propre terminologie et il convient de la connaître d'abord afin de comprendre leur principe :
Genèse : c'est la première phase de l'algorithme, il s'agit d'une population initiale de taille N.
Chromosome : c'est une chaîne représentant les caractéristiques de l'individu.
Phénotype : c'est un ensemble de paramètres ou une structure décodée.
Evaluation : c'est la phase de calcul de la fonction de fitness.
Sélection : c'est le choix des individus qui vont se reproduire.
Croisement : c'est la phase de production des descendants.
Mutation : c 'est la modification d'un chromosome dans le but d'améliorer les caractéristiques de l'individu. [23]

2.4 Principe des algorithmes génétiques

Les algorithmes génétiques dépendent de trois opérations principales : la sélection, le croisement et la mutation. Ces opérations seront effectuées sur une population composée d'individus. Le résultat de l'application des trois opérations est une génération à partir de laquelle un meilleur individu sera choisi parmi les autres.
Lorsque la population est choisie, chaque individu passe une évaluation (également appelée fitness) où il recevra un score d'adaptation à son environnement. Ce score sera pris en compte lors de l'application des trois opérations sur les individus.
Dans l'opération de sélection, les individus ayant un faible score d'adaptation seront éliminés ; seuls les individus avec le score d'adaptation élevé resteront dans la population où ils seront divisés en couples, et chaque couple passera par l'opération de croisement, donnant ainsi un nouvel individu. Ce nouvel individu passera l'opération de mutation où l'un de ses chromosomes sera modifié au hasard. Après la mutation, ces nouveaux individus formeront une nouvelle génération qui remplacera la première et ils passeront par l'étape d'évaluation et les opérations suivies.

Les deux étapes Le croisement et la mutation peuvent également être appelés reproduction. Le processus sera arrêté lorsque la génération appropriée sera obtenue, laquelle sera déterminée par une condition spécifique ; l'individu choisi sera celui qui a le meilleur score d'adaptation.
La mise en'uvre d'un algorithme génétique peut être résumée comme suit :

- ✓ Création d'une population initiale.
- ✓ Evaluation des individus de la population.
- ✓ Sélection des meilleurs individus.
- ✓ Reproduction (croissement et mutation).
- ✓ Formation d'une nouvelle génération. [24]

La figure suivante résume les étapes d'un algorithme génétique :

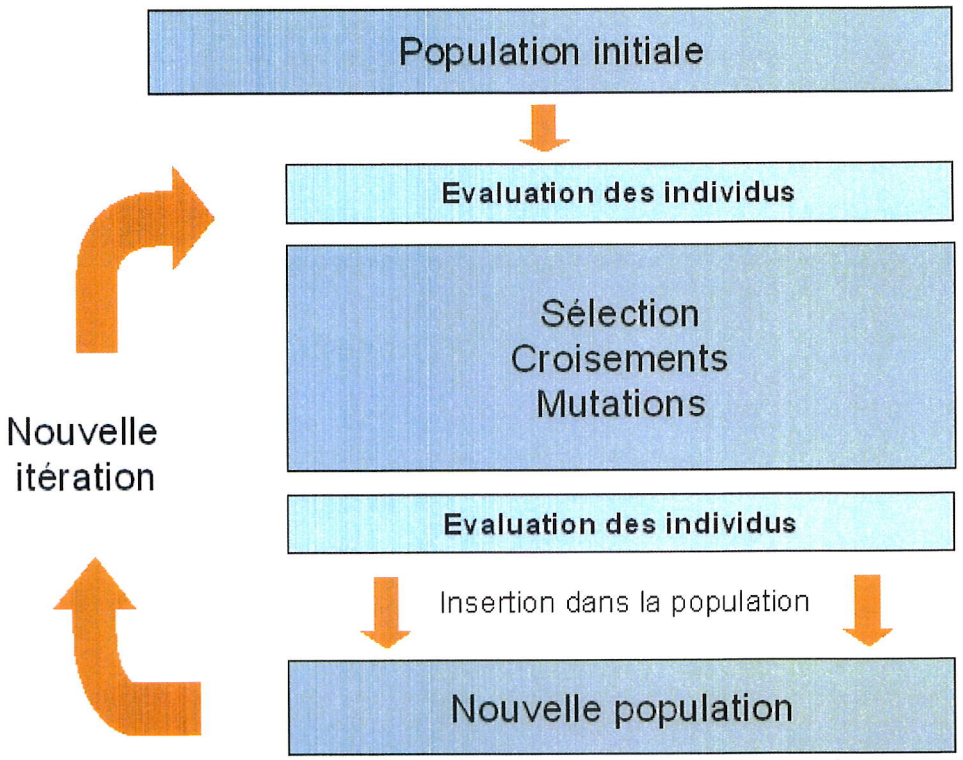


FIGURE 2.1 – Processus générale d'un algorithme génétique [30]

Pour utiliser des algorithmes génétiques afin d'obtenir une solution optimale à un problème spécifique, certains éléments devraient être disponibles :

- ✓ La population initiale sera choisie en suivant une méthode spécifique.
- ✓ Les individus de la population seront codés par l'un des types de codage décrits dans ce qui suit.

- ✓ La fonction d'évaluation, également appelée fitness, est celle qui sera optimisée.
- ✓ Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état. [24]
- ✓ Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation. [24]

2.4.1 Création de la population

Une population qui a N individus est créée en suivant une méthode spécifique sur un espace particulier et il y a m variables qui seront optimisées. Chaque chromosome d'un individu possède m gènes :

$$I = (x_1^1, x_2^1, \cdots\cdots\cdots, x_m^1)$$

Par conséquent, la population peut être représentée comme suit :

$$pop = \begin{cases} I_1 = (x_1^1, x_2^1, \cdots\cdots\cdots\cdots & x_m^1) \\ I_2 = (x_1^2, x_2^2, \cdots\cdots\cdots\cdots & x_m^2) \\ I_2 = (x_1^3, x_2^3, \cdots\cdots\cdots\cdots & x_m^3) \\ \vdots & \vdots \\ I_2 = (x_1^n, x_2^n, \cdots\cdots\cdots\cdots & x_m^n) \end{cases} [25]$$

Chaque chromosome qui a m gènes compose un individu I et leurs valeurs sont choisies au hasard.
Le choix de la population initiale peut contrôler la vitesse de l'algorithme génétique.

2.4.2 Le codage

Avant l'application des trois opérations (sélection, croisement, mutation), les individus de la population doivent être représentés sous une forme spécifique afin de faciliter le travail des opérations suivantes. Cette opération est appelée codage, où chaque chromosome des individus sera représenté à l'aide d'un type spécifique de codage. Ces types sont :

2.4.2.1 Codage binaire

Dans ce type, les chromosomes de chaque individu sont représentés par une chaîne de 0 et de 1. C'est l'une des formes de codage les plus utilisées. La figure suivante montre un exemple de codage binaire :

Chromosome A	10110010110010101110010 1
Chromosome B	11111110000011000001111 1

FIGURE 2.2 – Exemple d'un codage binaire

2.4.2.2 Codage octal

Dans cette forme de codage, chaque chromosome des individus est représenté en nombres octaux (0-7). La figure suivante montre un exemple de codage octal :

Chromosome A	57412361
Chromosome B	26345170

FIGURE 2.3 – Exemple d'un codage octal

2.4.2.3 Codage hexadécimal

Les chromosomes de chaque individu sont représentés en hexadécimal où la représentation sera (0-9, A-F). La figure suivante montre un exemple de codage hexadécimal :

Chromosome A	56AB
Chromosome B	39CF

FIGURE 2.4 – Exemple d'un codage hexadecimal

2.4.2.4 Codage par permutation

Dans ce type de codage, chaque chromosome des individus est représenté par une série de nombres ou de symboles qui ne peuvent pas être répétés, et il est placé dans une séquence. Les figures suivantes montrent un exemple de codage par permutation avec des nombres et des symboles :

Chromosome A	1 6 8 4 5 9 7 2 3
Chromosome B	9 6 8 5 4 7 2 3 1

FIGURE 2.5 – Exemple d'un codage par permutation avec des nombres

2.4.2.5 Codage par valeur

Dans le codage par valeur, chaque chromosome est représenté comme la chaîne d'une certaine valeur. La valeur peut être un entier, un nombre réel, un caractère ou un objet.[26] La figure suivante montre un exemple de codage par valeur :

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

FIGURE 2.6 – Exemple d'un codage par valeur

2.4.2.6 Codage par arbre

En encodage arborescent, chaque chromosome est une arborescence d'objets, tels que des fonctions ou des commandes en langage de programmation.[26]
La figure suivante montre un exemple d'encodage en arborescence :

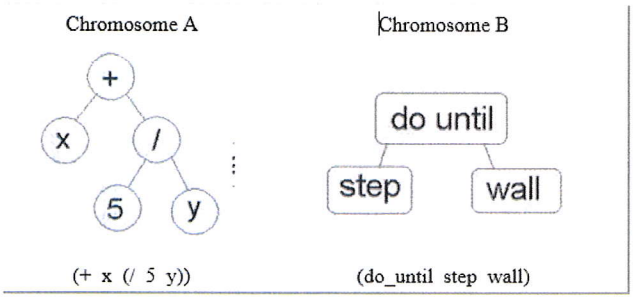


FIGURE 2.7 – Exemple d'un Codage par arbre

2.4.3 Fonction d'évaluation (fitness)

La fonction d'évaluation est celle qui sera optimisée soit maximisée, soit minimisée. Avec l'utilisation de la fonction d'évaluation, chaque individu obtiendra un score d'adaptation à son environnement. La fonction d'adaptation peut affecter directement la qualité des résultats obtenus par l'AG ainsi que le temps d'exécution. [24] Etant donné une fonction f réelle à une ou plusieurs variables, le problème d'optimisation sur l'espace de recherche E s'écrit de la manière suivante :[24]

$$max_{x \in E} f(x)$$

Dans beaucoup de problèmes, l'objectif est exprimé sous forme de minimisation d'une fonction coût h : [24]

$$min_{x \in E} h(x)$$

Le passage du problème de minimisation à un problème de maximisation est obtenu par transformation de la fonction h selon la relation suivante :[24]

$$f(x) = \frac{1}{1 + h(x)}$$

2.4.4 La sélection

Dans cette étape, les individus qui seront choisis pour la reproduction sont ceux qui ont un score d'adaptation élevé. Ces individus sont ceux qui possèdent les meilleurs chromosomes et peuvent reproduire une meilleure génération. Ils sont considérés comme les parents de la nouvelle génération qui remplacera l'ancienne.
Pour obtenir les individus qui procéderont aux opérations suivies, il existe plusieurs types de sélection que l'un d'eux peut être choisi :

2.4.4.1 Sélection de la roue de roulette

Dans ce type de sélection, les individus qui ont un score élevé d'adaptation sont ceux qui ont une probabilité élevée d'être choisis.
Selon cette méthode, chaque chromosome sera dupliqué dans une nouvelle population proportionnellement à sa valeur d'adaptation. On effectue, en quelque sorte, autant de tirages avec remises qu'il y a d'éléments dans la population.[27]
La probabilité de sélection d'individu x_i est appelée $p(x_i)$ est définie comme suit :

$$p(x_i) = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)}$$

Où :
N : taille de la population
f(x_i) : Fitness de l'individu

2.4.4.2 Sélection par tournoi

Dans ce type, un nombre spécifique d'individus est choisi au hasard afin de participer à une compétition dans laquelle l'individu ayant le meilleur score d'adaptation est celui qui va gagner et il fera partie de la population qualifiée. Cette opération prend fin lorsqu'un nombre spécifique d'individus est obtenu, et ce sont eux qui passent par le reste des étapes.
Également,Les individus qui participent à un tournoi sont remis ou sont retirés de la population, selon le choix de l'utilisateur.[24]
Cette méthode offre la possibilité de l'existence d'un individu avec un faible score d'adaptation, ce qui peut donner une variété dans les chromosomes de la population qualifiée.

2.4.4.3 Sélection par rang

Cette méthode dépend de l'ordre des individus du meilleur au pire. Après cela, chaque individu se verra attribuer un numéro représentant son rang. Par exemple, le meilleur individu à un rang de N où N est la taille de la population et le plus mauvais individu a un rang de 1. Ensuite, le choix des individus sera contrôlé par la probabilité indexée sur le rang des individus, définie comme suit :[23]

$$La probabilité de sélection(Parent_i) = \sum_{je me population} (Rang(Parent_i)/Rang(Parent_j))$$

2.4.4.4 Sélection uniforme

Dans ce type, la sélection des individus est faite de manière aléatoire sans prendre en compte le score d'adaptation ; cela signifie que les individus ayant un faible score d'adaptation ont les mêmes chances d'être sélectionnés que les individus ayant un score élevé d'adaptation. La probabilité que l'individu soit sélectionné est définie comme suit :

$$p(x_i) = \frac{1}{N}$$

Où :
x_i : L'individu de la population.
N : la taille de la population.

2.4.5 Croisement

Après la sélection, la population qualifiée passera par le croisement où de nouveaux individus ou enfants seront obtenus des parents. Dans cette étape, les individus seront choisis au hasard pour former des couples. Les parents de chaque couple doivent échanger des parties de leurs chromosomes pour former de nouveaux individus (enfants). Cette étape est contrôlée par la méthode de croisement choisie et la probabilité de croisement.

2.4.5.1 Croisement en un point

Dans cette méthode, le croisement entre les chromosomes des parents a lieu en un point choisi de manière aléatoire, mais a la même position dans les deux et divise les chromosomes en deux parties. Après cela, une permutation se produit lorsque la première partie du premier parent est combinée à la seconde partie du deuxième parent pour produire le premier enfant. Alors que le deuxième enfant est le résultat de la combinaison de la première partie du deuxième parent et de la deuxième partie du premier parent, comme expliqué dans la figure suivante :

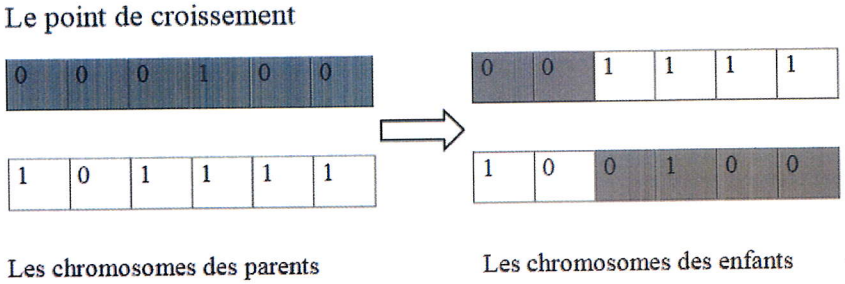


FIGURE 2.8 – Exemple d'un croisement en un point

2.4.5.2 Croissement en deux points et croissement en k-points

Dans ce type, le croisement entre les chromosomes des parents a lieu en deux points choisis au hasard, mais qui ont la même position dans les deux et divisent les chromosomes en trois parties. Le premier enfant sera le résultat de la combinaison de la première, et troisième parties du premier parent avec la deuxième partie du deuxième parent, cette partie étant située entre les deux points. La deuxième partie du premier parent, située entre les deux points, sera combinée avec les première et troisième parties du deuxième parent pour donner naissance au deuxième enfant, comme expliqué dans la figure suivante :

Ce processus peut également être effectué en croisement de k-points, où les chromosomes des parents seront divisés en parties en un nombre spécifique de points (k).

1.5 Conclusion

De nos jours, la cryptographie est utilisée dans presque tout ce qui a un rapport avec la vie privée afin d'assurer la confidentialité qui est une question importante et sensible pour les personnes du monde entier et de confirmer l'intégrité des informations échangées. Dans les administrations, la cryptographie est utilisée pour protéger les informations personnelles et privées du citoyen.De même, les communications en ligne sont protégées par la cryptographie afin de sécuriser les contacts et l'échange d'informations entre les utilisateurs. Les sites de vente en ligne utilisent également la cryptographie pour sécuriser les informations personnelles et financières de leurs clients, etc

Pour cette raison, le développement dans le domaine de la cryptographie ne peut pas être arrêté et de nouvelles méthodes sont toujours suggérées.

L'utilisation de la cryptographie peut être étendue à un autre niveau en utilisant d'autres méthodes qui cherche à optimiser la résolution du problème parmi lesquelles nous pouvons distinguer celle des algorithmes génétiques exposée dans le suivant chapitre.

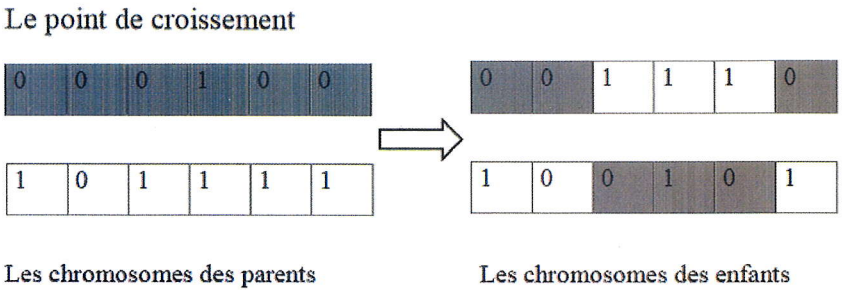


FIGURE 2.9 – Exemple d'un croissement en deux points

2.4.5.3 Croisement uniforme

Dans ce type, le croisement se produit dans des positions aléatoires sauf que les positions seront les mêmes dans les deux parents et que les chromosomes ne sont pas divisés en parties ; chaque valeur des chromosomes est traitée indépendamment. Les enfants obtenus seront la permutation entre les valeurs des chromosomes des parents dans ces multiples positions, comme expliqué dans la figure suivante :

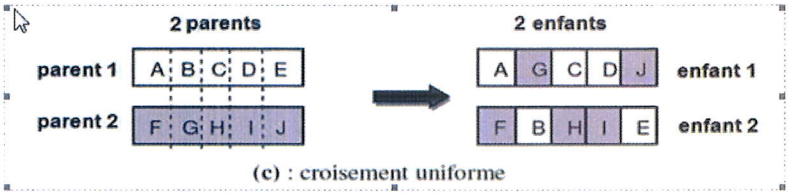


FIGURE 2.10 – Exemple d'un croissement uniforme

2.4.6 Mutation

Le rôle de cet opérateur est de modifier aléatoirement, avec une certaine probabilité, la valeur d'un composant de l'individu. [27] Cette modification de la valeur du chromosome produit un nouvel individu. Après l'application de cette étape, il en résulte une nouvelle population présentant de nombreuses caractéristiques différentes et de meilleures caractéristiques que la population initiale.
La figure suivante représente un exemple d'opérateur de mutation :

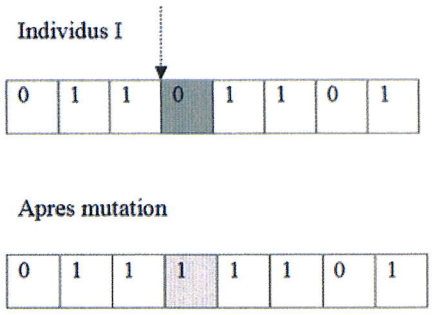


FIGURE 2.11 – Exemple de mutation

Il existe trois types de mutation :

2.4.6.1 Mutation en un point

Dans ce type, la mutation se produit dans une valeur du chromosome.

2.4.6.2 Mutation en deux points ou multipoints

Dans ce type, la mutation se produit en deux valeurs ou en plusieurs valeurs du chromosome.

2.4.6.3 Mutation par valeur

Dans ce type, la mutation se produit entre les valeurs du chromosome. La figure suivante représente des exemples concernant les types précédents :

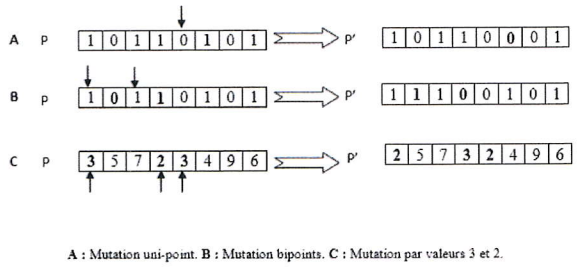


FIGURE 2.12 – .Exemples sur la mutation

2.5 Les avantages et les limites des algorithmes génétiques

2.5.1 Les avantages des algorithmes génétiques

Les algorithmes génétiques sont utilisés pour introduire des solutions optimales aux différents problèmes dans différents domaines. Ils sont utilisés dans les domaines de l'économie et de la finance, de la chimie, des applications de jeux, de l'informatique, etc. De plus, les algorithmes génétiques sont utilisés pour obtenir la solution optimale à de nombreux problèmes connus tels que le voyageur de commerce.
L'utilisation des algorithmes génétiques dans de multiples domaines est née de ses multiples avantages, dont certains seront mentionnés dans les lignes suivantes :

•Tout d'abord, le concept de l'algorithme génétique est très simple et peut être compris et mis en oeuvre.

• Les algorithmes génétiques lancent la recherche d'une solution optimale parmi un groupe de solutions et non une solution unique.

•La probabilité de croisement et de mutation permettent parfois d ?éviter de tomber dans un optimum local et se diriger vers l'optimum global. [23]

•Les algorithmes génétiques utilisent des règles probabilistes pour trouver la solution optimale, pas des règles déterministes, qui offrent davantage de diversité.

•Les algorithmes génétiques peuvent être facilement utilisés en mode parallélisme.

•Sur des problèmes mixtes discrets / continus, les algorithmes génétiques fonctionnent très bien.

2.5.2 Les limites des algorithmes génétiques

Bien que les algorithmes génétiques présentent de nombreux avantages, ils ont leurs inconvénients et leurs limites comme toute autre méthode. Certaines de ces limitations sont les suivantes :

•Les algorithmes génétiques sont lents et peuvent être coûteux en calcul.

•Le choix d'une fonction d'évaluation qui représentera le problème peut être difficile.

•Le choix des autres paramètres de l'algorithme génétique, tels que la taille de la population, le taux de croisement et le taux de mutation doit être fait avec beaucoup de soin.

•Dans le processus de génération de nouvelles solutions, un problème peut survenir et il doit être pris en compte avant de commencer l'application de l'algorithme génétique, qui est la convergence prématurée. Ce problème peut survenir lorsqu'un individu est plus en forme que les autres. Cet individu peut reproduire plus d'individus, ce qui limite la diversité de la nouvelle population. par conséquent, l'algorithme génétique fait converger l'optimum local qui représente cet individu et ne recherche pas l'optimum global qui devrait être la solution optimale pour la fonction d'évaluation.

2.6 Algorithmes génétiques parallèles

Au cours des dernières années, les chercheurs ont proposé des méthodes d'algorithmes génétiques parallèles afin d'obtenir des solutions optimales à plusieurs problèmes, et ce pour deux raisons : les limites des algorithmes génétiques séquentiels qui ont été vues dans la sous-section précédente, et à cause du fait que les algorithmes génétiques peuvent être facilement utilisés dans un mode parallèle.
Les algorithmes génétiques parallèles suivent les mêmes étapes que les algorithmes génétiques séquentiels ; sauf que l'exécution de plusieurs instances est faite en même temps pour introduire une solution optimale à un seul problème. Ceci est réalisé en tirant parti du processeur multic'ur qui permet d'exécuter plusieurs tâches simultanément.
Par conséquent, les algorithmes génétiques parallèles résultent de la parallélisation d'une étape de l'algorithme génétique ou de la parallélisation de tout l'algorithme génétique.
Autrement dit, les algorithmes génétiques parallèles sont utilisés parce qu'ils réduisent le temps de calcul et améliorent la qualité des individus, car ils offrent la possibilité d'obtenir de plus en plus d'individus différents dont les caractéristiques varient ; par conséquent, la solution optimale qui en résultera peut être plus satisfaisante que la solution optimale obtenue par un algorithme génétique séquentiel. En outre, les algorithmes génétiques parallèles peuvent limiter les problèmes de performances rencontrés dans les algorithmes génétiques séquentiels.
L'application de l'algorithme génétique peut être réalisée en suivant l'un des modèles de schémas de parallélisation existants résumés à travers la figure suivante [28].

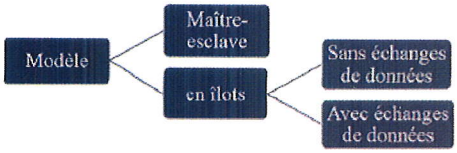


FIGURE 2.13 – Classification des modèles de parallélisation des algorithmes génétiques

2.6.1 Modèle maître esclave

Ce modèle est divisé en deux parties. La première partie est le maître et la seconde partie est l'esclave. Le processus maître est celui qui possède la totalité de la population et exécute les opérations de base (sélection, croisement, mutation) sur la population globale. Après cela, les individus résultants sont envoyés du processus maître à un nombre spécifique de processus esclaves qui calculent leur fitness et renvoient les résultats au processus maître.
Le modèle maître-esclave fonctionne comme l'algorithme génétique séquentiel, et le travail parallélisé est effectué à l'étape du calcul de fitness des individus résultants.
Ce modèle possède plusieurs avantages tel que la facilité d'implémentation et l'exploration du même espace de recherche que la version séquentielle et ce, plus rapidement[29]. Malgré cela, il présente aussi certains inconvénients. En particulier, il n'est pas extensible et il surcharge le réseau par les communications dans le cas d'un grand nombre de processeurs.[28]
La figure suivante montre un schéma du modèle maitre-esclave :

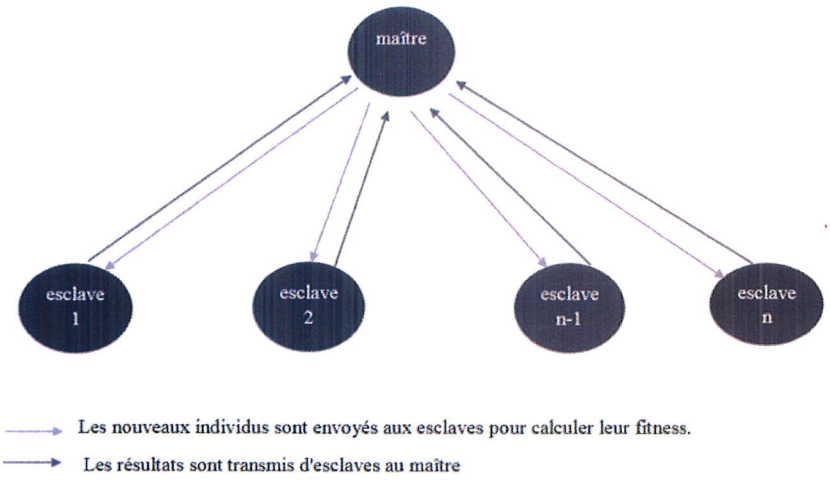


FIGURE 2.14 – le modèle maître-esclave

2.6.2 Modèle en ilots

Certaines personnes peuvent percevoir le monde comme la division d'une énorme population en petites populations vivant sur les différents continents où elles évoluent indépendamment. Cependant, il y a aussi le fait que certains individus d'une population spécifique peuvent migrer de leur population vers une autre où ils participent à l'étape de l'évolution de leur nouvelle population. Le résultat de cette participation est la diversité des nouveaux individus. L'application de cette idée aux algorithmes génétiques parallèles a produit les modèles en îlots.
Ce modèle est l'exécution de plusieurs algorithmes génétiques séquentiels en même temps. Chacun de ces algorithmes génétiques a sa propre population, mais ils travaillent afin d'obtenir une solution optimale à un problème unique.
Les modèles d'îlots peuvent être divisés en deux types : sans migration, avec migration.

2.6.2.1 Modèle en ilots sans migration

Dans ce type de modèle d'îlots, la population est divisée en plusieurs sous-populations, et chacune des sous-populations constituera la population initiale d'un algorithme génétique séquentiel. Cela signifie que ces algorithmes génétiques séquentiels rechercheront la solution optimale parmi ces sous-populations en parallèle. Le terme sans migration signifie que les sous-populations sont indépendantes les unes des autres.
En fin de compte, la solution optimale sera le meilleur individu obtenu parmi les meilleurs individus des différentes sous-populations.
La figure suivante montre le schéma du modèle d'île sans migration :

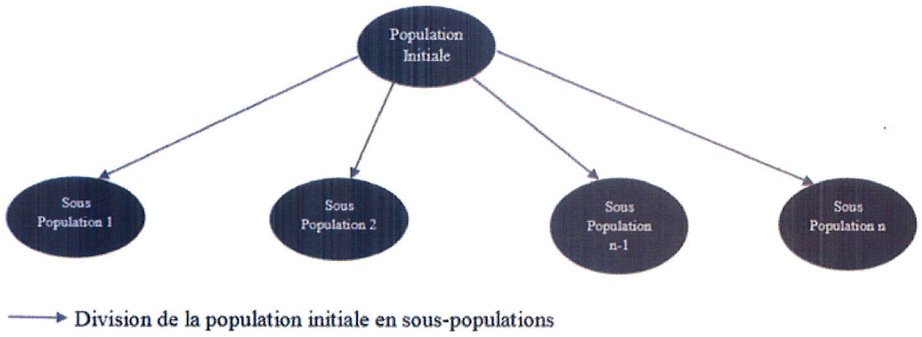


FIGURE 2.15 – le modèle en îlots sans migration

2.6.2.2 Modèle en îlots avec migration

Dans ce type de modèle d'îlots, il existe un échange d'informations entre les sous-populations et l'algorithme génétique séquentiel est exécuté sur chacune d'entre elles en parallèle. L'échange d'informations est la migration de certains individus d'une sous-population à une autre après un certains nombre d'itérations. Les individus qui migreront peuvent être les meilleurs ou les pires de leur population, ou ils sont sélectionnés au hasard et remplaceront d'autres individus (meilleurs, pires, aléatoires) dans leur nouvelle population. Cette migration des individus offre plus de diversité pour le calcul des nouveaux individus qui en résulteront. Les populations qui échangeront les individus sont choisies en utilisant une topologie logique, tandis que les individus sont envoyés en utilisant l'une des stratégies d'échange expliquées ci-après.
La figure suivante montre le schéma du modèle d'île avec migration :

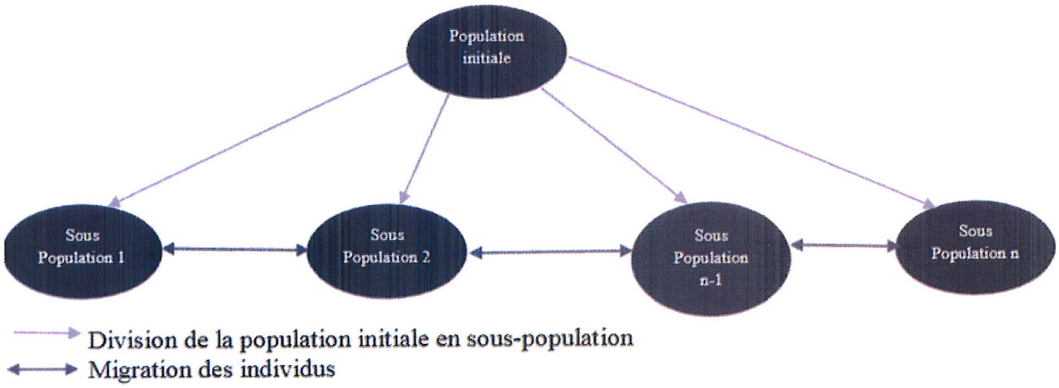


FIGURE 2.16 – Le modèle en îlots avec migration

Dans ce type de modèle, une migration en redistribution peut être appliquée. Dans ce cas, il y a deux parties : la première partie est le maître qui est celui qui va redistribuer la population initiale en sous-populations et les envoyer à la seconde partie où il y a plusieurs algorithmes génétiques séquentiels, et chacune des sous-populations est la population initiale de l'un d'entre eux et chaque population est indépendante de l'autre. Après un certain nombre d'itérations, la génération résultante de chaque algorithme génétique sera renvoyée au maître. Ce maître redistribuera à nouveau cette nouvelle population en sous-populations et les renverra à la deuxième partie. Ce travail sera effectué plusieurs fois jusqu'à ce que la condition d'arrêt soit atteinte.
La figure suivante résume cette situation (modèle d'îlots avec migration en redistribution).

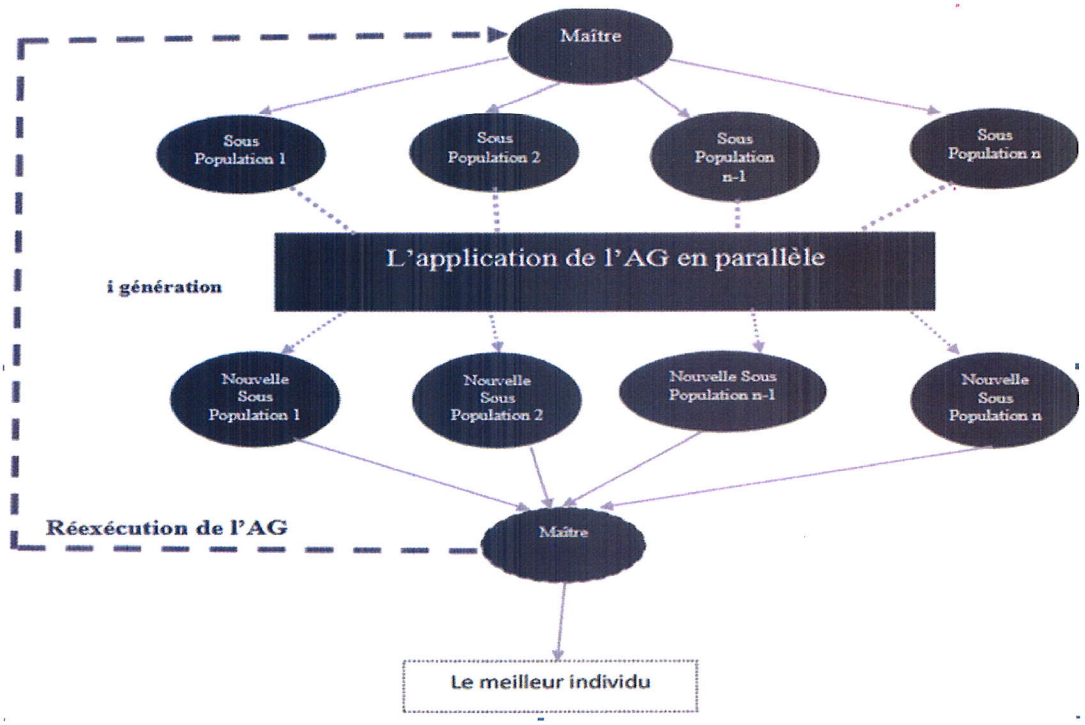


FIGURE 2.17 – le modèle en îlots avec migration en redistribution

2.6.3 Les stratégies d'échange

2.6.3.1 Topologie en anneau

Dans cette topologie, chaque processeur envoie ses meilleurs individus vers le processeur voisin de prochain rang pour substituer les individus les plus mauvais de sa population puis , ce dernier envoie les meilleurs de la nouvelle génération au prochain processeur dans l'anneau. Le processus de migration peut se faire à la fin de chaque itération. [28]
La figure suivante présente la topologie en anneau :

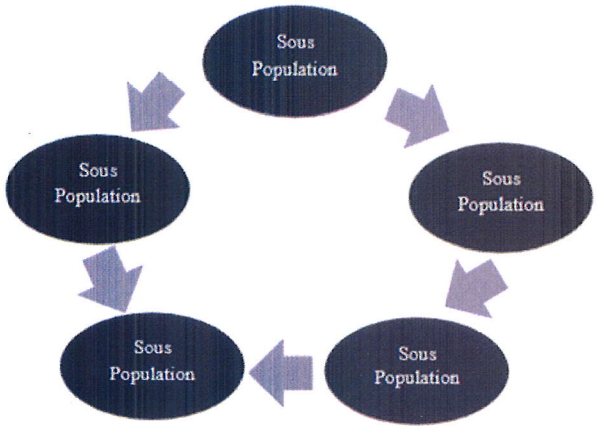


FIGURE 2.18 – Topologie en anneau

2.6.3.2 Topologie aléatoire

Dans cette topologie, chaque processeur envoie ses meilleurs individus vers un autre processeur dont le rang est calculé aléatoirement [28] comme le montre la figure suivante.

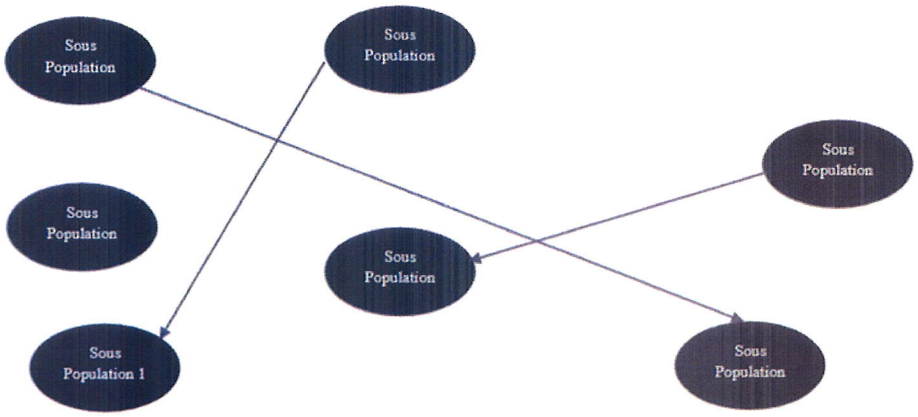


FIGURE 2.19 – Topologie aléatoire

2.7 Conclusion

Les algorithmes génétiques sont en développement constant. Beaucoup de recherches sont faites sur ce domaine. De nombreuses méthodes d'algorithmes génétiques sont proposées afin de résoudre des problèmes connus que les algorithmes déterministes ne pourraient pas résoudre.

Le choix d'un modèle d'algorithme génétique spécifique dépend de la nature du problème et du résultat attendu. Également le choix d'un type spécifique qui sera exécuté dans les opérations de sélection ; de croisement et de mutation doivent être bons pour éviter le problème de la convergence prématurée.

Les algorithmes génétiques séquentiels et les algorithmes génétiques parallèles ont leurs propres avantages et limites. Pour cette raison, le choix de l'un d'entre eux, à savoir celui qui obtient la solution optimale à un problème spécifique, doit être adapté aux besoins de l'application. De plus, le temps d'exécution de l'algorithme génétique choisi (séquentiel ou parallèle) devrait être raisonnable.

Chapitre 3

Conception

3.1 Introduction

Dans les deux chapitres précédents, le concept de cryptographie et d'algorithme génétique a été clairement introduit afin de faciliter la compréhension de ce chapitre et du suivant.
Dans ce chapitre, une explication des étapes à suivre pour obtenir un cryptage sécurisé d'une image sera présentée. Et pour obtenir ce cryptage, un algorithme génétique sera utilisé et il comprend : création de population initiale, reproduction (croisement, mutation), évaluation et sélection. Ces étapes seront répétées pour un nombre spécifique d'itérations. Après cela, la solution optimale sera obtenue représentant le cryptage d'une image donnée.
Pour résoudre ce problème, à savoir comment obtenir un cryptage sécurisé d'une image, utiliser un algorithme génétique peut être considéré comme l'une des méthodes de résolution prometteuse à ce problème. L'application de l'algorithme génétique a pour objectif d'obtenir la différence maximale entre l'image originale et l'image chiffrée correspondante ; chose garantie par les étapes de l'algorithme génétique.
Comme c'était présenté dans le chapitre précédent, l'algorithme génétique peut être utilisé en mode séquentiel ou en mode parallèle. L'application d'un algorithme génétique séquentiel peut s'avérer très gourmande en temps de calcul ou en ressources requises surtout pour des problèmes traitant des images. C'est pourquoi l'objectif de notre travail était le développement d'un algorithme génétique parallélisant le principe de cryptage génétique basé sur les positions adopté par l'algorithme PosESecL2 [31]. Pour cet objectif, une première alternative de parallélisassions a été proposée dans[32]exploitant un modèle dit en îlots sans migration (c'est à dire, sans échange de paramètres entre les modules parallèles) où deux nouveaux algorithmes de cryptage ont été proposés,PFSPos et PVSPos, adoptant, respectivement, une segmentation fixe et variable du codage à base de positions de PosESecL2.
Notre travail représente une continuité à ce travail, en testant cette fois-ci, une parallélisassions suivant le même modèle en îlots mais avec échange de paramètres entre les modules parallèles. Ainsi, notre travail portera sur le développement d'un nouvel algorithme de cryptage parallèle exploitant le modèle en îlots avec migration.

3.2 Formalisation du problème de chiffrement

Étant donnée une image originale I constituée de (n*m) pixels. Le résultat de cryptage de I sera une image appelée, par exemple, I' constituée de (n*m) pixels, c'est à dire de même taille mais de contenu différent, bien évidemment. Comme I' est de contenu illisible ou non compréhensible, une opération de déchiffrement sera nécessaire pour aboutir au contenu en clair de l'image. Dans ce cas, seul les possesseurs du bon paramètre de déchiffrement, appelé clé de déchiffrement, peuvent avoir accès au contenu lisible.
Comme c'était mentionné précédemment, l'objectif visé par notre conception de cryptage (cryptage génétique) est d'obtenir la différence maximale entre l'image originale et la version chiffrée correspondante. Pour cette raison, une fonction F mesurant la différence entre les deux images (originale et cryptée) sera utilisée. Quand F atteint des valeurs maximales, l'algorithme converge en retournant l'image équivalente au chromosome codant la solution calculée.
La figure suivante rappelle les paramètres du processus de cryptage.

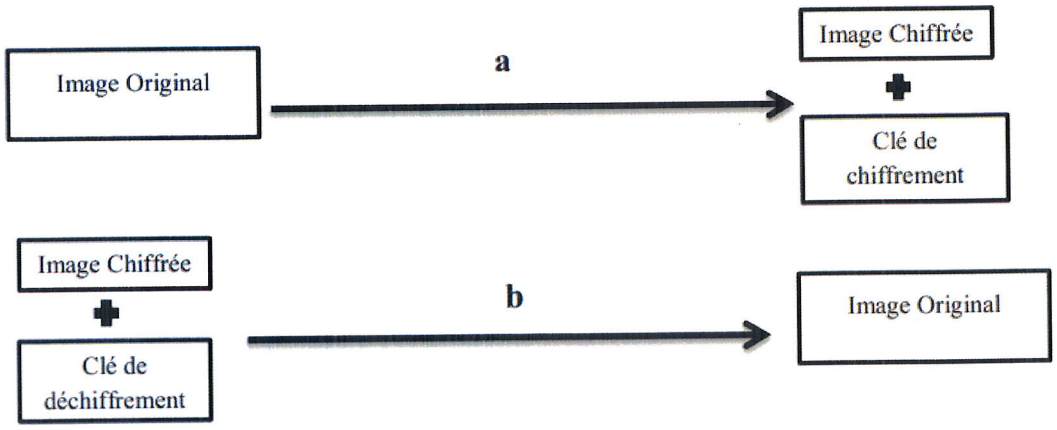


FIGURE 3.1 – a- Processus de chiffrement, b- processus de déchiffrement

3.3 Parallélisation en îlots de PosESecL2 avec migration

3.3.1 Les algorithmes PosESecL2, PFSPos et PVSPos

PosESecL2, abréviation pour Position Based Evolutionary Secure Level 2, est un algorithme génétique de cryptage d'images opérant sur les positions des éléments codant une image. Son fonctionnement est illustré à travers l'algorithme ci-dessous résumant les différentes étapes opératoires menant au résultat (image chiffrée) partant du codage d'une image en entrée (image en clair).

Algorithm 3.1 PosESecL2

DÉBUT

1)- Codage de l'image en clair (chromosome initial).

2)- Création de la population initiale.
Tant que Critère d'arrêt non satisfait **faire**

3)- Application des opérateurs de reproduction (croisement / mutation) pour obtenir de nouveaux individus (enfants).

4)- Évaluation des individus.

5)- Sélection des individus qui vont survivre pour la prochaine génération.
Fin tant que

6)- Calcul de la clé de chiffrement (Clé de session).
FIN

Pareil à tout algorithme génétique, la première étape de PosESecL2 est celle de codage.C'est, sans doute, l'étape la plus importante.
Vu que l'espace RVB de codage d'images est celui utilisé, l'ensemble des gènes d'une chromosome est formé des valeurs des composantes R, V et B codant les différents pixels d'une image. Ainsi, pour une image de taille (n*m),le chromosome codant cette image aura une taille égale à (n*m*3).
La figure suivante présente une illustration du mode de codage proposé dans PosESecL2.

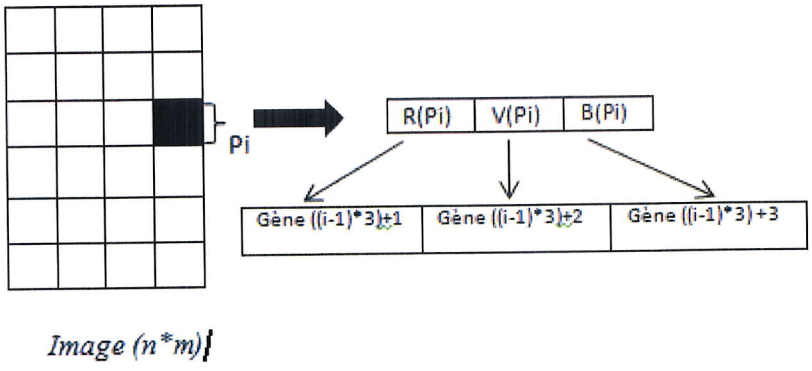


FIGURE 3.2 – Codage adopté par PosESecL2.

Comme PosESecL2 est un algorithme génétique séquentiel, donc, le principal problème de tels algorithmes reste, principalement, le grand temps d'exécution. En effet, le temps de calcul par PosESecL2 augmente proportionnellement avec la taille des images soumises au chiffrement. C'est clair que d'après le codage adopté, le temps de calcul correspondant à des images de grandes tailles sera considérablement plus grand que celui d'images de petites tailles.
Dans ce cas et pour améliorer le temps de calcul d'un tel mode de cryptage tout en bénéficiant du haut niveau de sécurité de PosESecL2, une parallélisation sera suggérée.
La première solution proposée dans [32], consiste à segmenter (diviser) le chromosome initial codant l'image originale, pour donner lieu à plusieurs sous-chromosomes de tailles sécuritaires, chacun évoluant indépendamment et en parallèle des autres suivant le même principe de PosESecL2. Ainsi, la solution finale s'obtient par regroupement des solutions partielles produites par les différents modules de calcul parallèle.
Pour résumer, il s'agit d'une parallélisation suivant le modèle en îlots mais sans migration (modules s'exécutant en parallèle et indépendamment et sans échange de paramètres) C'est clair que ceci améliore considérablement le temps de calcul séquentiel de PosESecL2.
Pour la segmentation, deux modes ont été mis en oeuvre. Un premier réalise une segmentation sécuritaire fixe, où tous les sous-chromosomes sont de mêmes tailles, ce qui a donné lieu à un premier algorithme appelé PFSPos pour une parallélisation de taille fixe. Le deuxième mode adopte une segmentation sécuritaire variable et les sous-chromosomes peuvent avoir différentes tailles. PVSPos était l'algorithme résultat de ce dernier mode de parallélisation.
Maintenant et à travers notre présent travail, nous testerons la deuxième branche du modèle en îlots où des échanges d'information entre les modules parallèles peuvent avoir lieu. Il s'agit de l'exploitation du modèle en îlots avec migration en se limitant au mode de segmentation fixe illustré ci-dessous.
Dans ce qui suit, nous présenterons les étapes explicatives résumant le principe de cryptage génétique adopté.

3.3.2 Codage et sous-codages

L'étape de codage est l'une des étapes les plus importantes de l'algorithme génétique car elle influence directement les individus de la population, ce qui affecte la qualité de la solution optimale.
Dans notre conception du mode parallèle, le chromosome initial résultant du codage de l'image originale sera divisé en sous-chromosomes, qui auront tous la même taille. Cette dernière sera définie ultérieurement dans l'étape expérimentale.
Notons que le codage adopté est le même que celui de PosESecL2 (c'est le même aussi que celui de PFSPos et PVSPos), la figure suivante présente une explication du passage de codage aux sous-codages de tailles fixes.

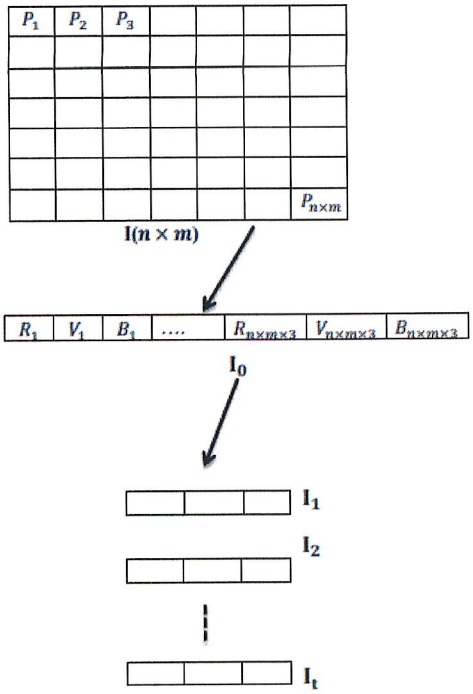


FIGURE 3.3 – Codage et sous-codages de tailles fixes.

Tels que :
I : est l'image originale,
I_0 :Chromosome initial (codage de l'image originale),
$I_{1,...,}I_t$:Sous- chromosomes initiaux résultant de la division suivant une taille fixe de I_0
t*l = n*m*3.

3.3.3 Création des populations initiales

Après l'étape de codage, la création de population est l'étape qui suit, où de nouveaux individus seront créés pour former la population initiale qui sera la première à passer par la reproduction et l'évaluation, pour aboutir à la sélection en dernier lieu.
Dans notre cas et vu que le chromosome initial représentant le codage de l'image originale donne lieu à t sous-chromosomes initiaux de mêmes tailles suite à une opération de division, donc t populations initiales seront crées chacune à partir d'un sous-chromosome en permutant aléatoirement des gènes des sous-chromosomes impliqués.
L'algorithme suivant résume les étapes de la création de sous-populations initiales.

Algorithm 3.2 Création d'une sous-population initiale

Début

Sous-population initiale = ∅

Répéter

1) permuter aléatoirement des gènes du sous-chromosome initial

2) ajouter l'individu résultant à la sous-population initiale.

Jusqu'à ce que le nombre d'individus spécifié soit atteint.

Fin

3.3.4 Opérateurs de reproduction

3.3.4.1 Croisement

Le croisement permet d'obtenir de nouveaux individus parmi les individus existants dans la population, ce qui améliore la qualité de la population. Cette étape sera effectuée en choisissant deux individus de la population et en appliquant le croisement sur eux, le résultat sera un ou deux nouveaux individus.

L'algorithme suivant illustre les étapes du croisement.

Algorithm 3.3 Croisement

Début

1- Sélection de deux individus au hasard.

2- Génération de deux points de croisement au hasard.

3- Application de l'opérateur de croisement choisi.

4- Ajout des enfants à la population.

Fin

Pour notre conception, l'opérateur de croisement choisis est le OX (Order Crossover) [33]dont les étapes se résument en :

1- choisir dans les deux parents une sous-séquence interne, comprise entre deux points de coupure tirés aléatoirement.

2- Recopier la sous-séquence interne du Parent1 dans le descendant Enfant1 aux mêmes positions et retirer du chromosome Parent2 les allèles compris dans cette sous-séquence.

3- Le chromosome Parent2 permet alors de former une séquence d'allèles résiduelle en partant du deuxième point de coupure et en considérant le chromosome comme une chaîne circulaire.

4- Compléter les gènes disponibles du descendant Enfant1 en lui transmettant dans l'ordre les allèles issus de la séquence résiduelle précédente.

La figure suivante donne un exemple applicatif de l'opérateur OX.

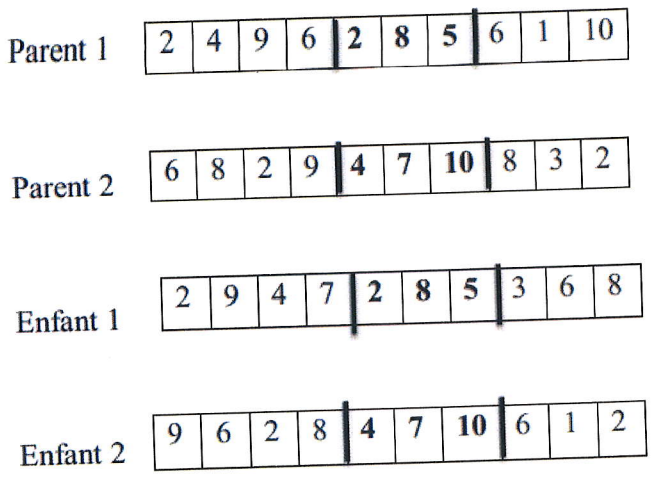


FIGURE 3.4 – Exemple sur l'opérateur OX .

3.3.4.2 Mutation

L'opérateur de mutation manipule les gènes d'un individu choisi au hasard. Dans ce cas, la mutation sera appliquée à l'individu en effectuant une permutation entre deux gènes de l'individu. L'individu résultant sera ajouté à la population.
L'algorithme suivant montre les étapes de l'opérateur de mutation.

Algorithm 3.4 Mutation
Début

1- Choisissez un individu au hasard dans la population.

2- Choisissez deux nombres au hasard qui représentent deux points c1 et c2, où 0 $\leq$c1, c2 $<$ taille de l'individu et c1 $\neq$c2.

3- Permutation entre l'élément à la position c1 et l'élément à la position c2.

4- Ajoutez le nouvel individu à la population.
Fin

3.3.5 Évaluation

L'étape de l'évaluation consiste à évaluer les individus de la population, chacun d'entre eux recevant un score d'évaluation afin de déterminer si ces individus ont une qualité bonne ou mauvaise. Cette opération sera effectuée à l'aide d'une fonction d'évaluation.
La fonction d'évaluation définie est la suivante :

$$F(I_i) = \sum_{j=1}^{n} \mid R_{ji} - R_{j0} \mid + \mid V_{ji} - V_{J0} \mid + \mid B_{ji} - B_{j0} \mid$$

Où : R_{ji} , V_{ji},B_{ji} et sont les valeurs des composantes R, V et B du $j^{éme}$ gène du $i^{éme}$ individu.
I_i est le $i^{éme}$ individu de la population.
n :est la taille des individus.
Cette fonction d'évaluation calcule la différence entre les valeurs d'un individu spécifique de la population et les valeurs du chromosome initial.
La valeur obtenue sera considérée comme le score d'évaluation de cet individu.

3.3.6 Sélection

Après l'évaluation, les individus sont maintenant prêts à passer par l'étape de sélection où les individus de bonne qualité seront choisis pour former la nouvelle population. L'opérateur choisi est celui de sélection par roulette sans possibilité de choisir plusieurs fois le même individu ; où les individus qui ont une bonne qualité auront plus de chances d'être choisis que les individus qui ont une qualité inférieure. Mais les individus de moindre qualité ont également la possibilité d'être choisis, ce qui préserve la diversité de la population résultante. Pour appliquer ce type de sélection, les étapes suivantes seront effectuées :

1- Calcul de « S » qui représente la somme de tous les scores d'évaluation des individus de la population.

2- Générez un nombre aléatoire « a » compris entre 0 et « S ».

3- Calcul de "Sum" qui représente la somme des scores d'évaluation. Le calcul s'arrête lorsque Sum est supérieur à "a".

4- l'individu correspondant au dernier score rajouté et faisant que Sum dépasse "a", cet individu sera sélectionné.

Algorithm 3.5 Sélection par roulette

N : nombre des individus qui seront choisis
Début
Tant que i $\leq$ N **faire**

1- Générez un nombre aléatoire "a".

2- Calculer la somme des scores d'évaluation « Sum » tant que « Sum » $\prec$ a .

3- Sélectionnez l'individu qui a le score d'évaluation où : a $\succ$Sum

4- Ajouter l'individu à la population.
Fin tant que
Fin

3.3.7 Migration

Comme il a été mentionné, ce travail s'inscrit dans la continuité du travail [32], qui consistait à utiliser le modèle d'îlots sans migration ; cependant. Dans le présent travail, l'étape de migration sera ajoutée.
La phase de migration consiste à faire migrer un individu (ou groupe d'individus) d'une population à l'autre afin d'améliorer la qualité de la population ciblée et d'obtenir une plus grande diversité de qualité de ses individus.
Une population peut contenir des individus de mauvaise qualité ou de bonne qualité. Ces deux types de qualité peuvent l'un dominer l'autre. En d'autres termes, les individus de bonne qualité peuvent être de nombre supérieur au nombre d'individus de mauvaise qualité et vice versa, ce qui conduit parfois à restreindre l'espace de recherche. C'est pourquoi et dans notre travail, l'étape de migration servira à réduire ce problème et à donner aux populations plus de diversité compte tenu des caractéristiques des générations résultantes.
Pour déterminer le type de migration dont la population a besoin avant d'exécuter l'opération de migration, chaque population après un certain nombre d'itérations doit être soumise à un test. Le résultat du test conclura sur le type de migration suggéré ; qui peut être soit une migration aléatoire, une migration du pire au meilleur ou finalement de meilleure au pire. Ces trois types seront expliqués en détail dans les sous-sections suivantes.
L'algorithme suivant présente les étapes du test de migration.

Algorithm 3.6 Test avant la migration

bon : le nombre d'individus de bonne qualité initialement vaut 0.
mauvais : le nombre d'individus de mauvaise qualité initialement égale à 0.
réponse : le résultat du test.
Début

1- Calculer « bon ».

2- Calculer « mauvais ».
Si bon> mauvais, **alors**
réponse =migration du pire au meilleur.
Sinon Si bon <mauvais **alors**
réponse = migration du meilleur au pire.
Sinon Réponse = migration aléatoire.
Fin si
Fin

3.3.7.1 Migration du pire au meilleur

Ce type de migration signifie que la population compte un nombre élevé d'individus de bonne qualité et un faible nombre d'individus de mauvaise qualité, ce qui signifie que les individus qui migreront de leur population vers celle-ci devraient être de mauvaise qualité afin d'obtenir un certain équilibre dans l'ensemble de qualités d'individus.

L'algorithme suivant illustre l'opération de la migration du pire au meilleur.

Algorithm 3.7 Migration du pire au meilleur

Début

1- Choisissez les pires scores d'évaluation d'une certaine population.

2- Énumérez les individus ayant les scores d'évaluation précédemment choisis.

3- Ajoutez ces individus à une deuxième population comportant globalement de bonnes caractéristiques (individus de bonnes qualités de nombre plus grand que celui d'individus de mauvaises qualités) et supprimez les de la première population.

Fin

3.3.7.2 Migration du meilleur au pire

Cela signifie que la population qui va accueillir d'individus comprend un nombre élevé d'individus de faible qualité et de nombre moindre d'individus de bonne qualité. En conséquence, les individus qui vont migrer de leur population devraient être de bonnes qualités. Il s'agit du cas contraire de la migration du pire au meilleur.
L'algorithme suivant présente les étapes d'une migration du meilleur au pire.

Algorithm 3.8 migration du meilleur au pire

Début

1- Choisissez les meilleurs scores d'évaluation d'une première population.

2- Cernez les individus correspondants aux scores de l'étape 1.

3- Ajoutez ces individus à une deuxième population de qualité opposée et supprimez-les de la première population.

Fin

3.3.7.3 Migration aléatoire

Le dernier type est basé sur le choix d'individus qui migreront de manière aléatoire, sans tenir compte du fait que ces individus soient de bonne ou de mauvaise qualité.
L'algorithme suivant montre les étapes de la migration aléatoire.

Algorithm 3.9 Migration aléatoire
Début

1- Choisissez des individus au hasard d'une première population.

2- Ajoutez ces individus à une deuxième population et supprimez-les de la première population.
Fin

Remarque :
Il est utile de noter que l'opération de migration ne peut pas être effectuée entre des populations contenant des individus d'un même type de qualités.Par exemple, si deux population peuvent participer au type de migration du meilleur au pire (c'est à dire toutes les deux comptent un grand nombre d'individus de bonnes qualités), dans ce cas, la migration entre elles est impossible car elles ont toutes deux besoin de diversifier leurs espaces de recherche en accueillant des individus de mauvaises qualités.
Cependant, si la première population peut participer au type de migration du meilleur au pire et la seconde population au type de migration du pire au meilleur,un échange d'individus (une migration d'individus) entre elles sera possible, car chacune complète le besoin de l'autre en terme de qualité d'individus.
En cas de non-existence d'une population de qualité opposée aux besoins d'une population donnée, une migration aléatoire sera exécutée.

3.3.8 Critère d'arrêt

Après l'exécution des étapes précédentes (croisement, mutation, évaluation, sélection et migration) pour un nombre spécifique d'itérations représentant le nombre de générations produites, l'exécution sera terminée lorsque le nombre d'itérations atteindra le nombre spécifié.

3.4 Clé de session et déchiffrement

Après avoir obtenu le chromosome codant l'image chiffrée, une clé de session sera générée afin de pouvoir inverser la procédure et obtenir l'image originale. L'image cryptée a les mêmes valeurs d'éléments que l'image originale, à la différence que ces éléments se trouvent dans des positions différentes de celles de l'image originale. En conséquence, la clé de déchiffrement contient les positions originales des gènes. Et comme le chiffrement, sur des instances différentes, d'une même image originale peut donner lieu à des images chiffrées différentes, donc, la clé de déchiffrement diffère d'une instance à une autre. De ce fait, on parle de clé de session.
Lors de l'étape de déchiffrement, la clé de session générée sera utilisée avec l'image chiffrée afin de placer les éléments du codage (gènes) dans leurs positions initiales pour obtenir l'image originale.

L'algorithme suivant décrit la phase de déchiffrement.

Algorithm 3.10 Déchiffrement

Clé : clé introduite par l'utilisateur.
Début
Si clé = bonne clé de session **alors**

1- Calcul du chromosome qui code l'image original.

2- Générez l'image originale à partir du chromosome calculé.
Fin si
Fin

3.5 Conclusion

Dans ce chapitre, les détails de la structure de ce travail ont été introduits et expliqués. Ce travail est considéré comme la continuité de celui [32] dans lequel le modèle d'îlots sans migration est utilisé afin d'introduire une nouvelle méthode de cryptage d'image utilisant un algorithme génétique et tirant parti de l'option de parallélisme afin de réduire le temps d'exécution.
Dans ce travail, le modèle d'îlots avec migration est exploité afin d'améliorer les résultats de cryptage en termes de qualité de l'image chiffrée et notamment de temps de calcul.

Chapitre 4

Expérimentation et Résultats

4.1 Introduction

Dans le chapitre précédent, nous avons présenté une description conceptuelle du processus de cryptage génétique que nous avons proposé afin de résoudre le problème du cryptage d'images. L'algorithme de base PosESecL2 qui est un algorithme génétique séquentiel de cryptage a été rappelé, en plus du principe de parallélisation de cet algorithme proposé par [32] et donnant lieu à deux algorithmes PFSPos et PVSPos.
Et comme notre travail représente une continuité au travail [32], où nous testerons l'application du même modèle de parallélisation qui est celui en îlots mais avec un échange d'information entre les modules parallèles (dit avec migration), il a été jugé utile de rappeler sur le présent chapitre, les valeurs de paramètres du mode de segmentation proposé dans [32] et que nous l'avons réutilisé (segmentation fixe).
Dans les sections qui suivent, nous illustrerons :

- ✓ La structure du programme qui réalise notre conception d'algorithme génétique parallèle de cryptage implémenté en JAVA,
- ✓ Réglage des paramètres de migration,
- ✓ Présentation des exemples de résultats expérimentaux et comparaison de l'algorithme développé avec au moins les trois algorithmes PosESecL2, PFSPos et PVSPos,
- ✓ Présentation des différentes fenêtres de l'application et de leurs composants.

4.2 La structure du programme

La description suivante présente les détails de la structure de l'application, où le travail de chacune des classes sera expliqué : Comme nous l'avons mentionné, le langage de programmation JAVA sera utilisé pour la mise en oeuvre de l'application. Cet environnement a été choisi car il est portable ; il n'a pas besoin de compilation spécifique pour chaque plate-forme et il reste indépendant de la machine où il sera exécuté.
En outre, JAVA a été choisi car c'est un langage de programmation multi-threading, ce qui signifie qu'il permet de créer un programme multi-threading et d'exécuter plusieurs tâches simultanément, chose qui répond parfaitement à notre objectif de parallélisation.

• **Classe Original _ image :** cette classe contient deux méthodes : la première est « **read _ image** » qui renvoie le chromosome initial qui représente l'image soumise au chiffrement (originale) selon le mode de codage proposé ; la seconde est « **chromo _ code** », cette méthode divise le chromosome initial en sous-chromosomes où chacun d'eux aura la même taille.

• **Classe Create _ pop :** cette classe contient une méthode qui crée des populations de manière aléatoire à partir des sous-chromosomes.

• **Classe Operands :** cette classe contient :

- **La méthode « crossover » :** cette méthode choisit deux individus aléatoirement et leur applique l'opération de croisement ; le résultat sera deux nouveaux individus et ils seront ajoutés à la population.
- **La méthode « mutation » :** un individu est choisi au hasard par cette méthode où une opération de permutation sera appliquée entre deux de ses gènes ; le résultat est un nouvel individu qui sera rajouté à la population.
- **La méthode « évaluation » :** le score d'évaluation de chaque individu sera calculé par cette méthode en suivant la fonction d'évaluation définie dans le chapitre précédent.
- **La méthode « sélection » :** cette méthode implémente l'application de la roulette en utilisant les scores d'évaluation calculés par la méthode précédente afin de sélectionner les individus qui formeront la nouvelle population.
- **La méthode « final _ évaluation » :** Une fois l'algorithme converge et l'individu "solution" sera déterminé, la méthode « **final _ évaluation** » sera utilisée pour calculer le score d'évaluation de cet individu.

•**Classe Migration :** cette classe contient des méthodes responsables de la définition du type de migration et de l'exécution de la migration entre les populations.

- **La méthode « migration » :** cette méthode renvoie le type de migration dont a besoin une population donnée (migration du pire au meilleur, migration du meilleur au pire, migration aléatoire).
- **La méthode « migration _ worst _ best » :** dans cette méthode, l'opération de migration sera appliquée entre deux populations, où les pires individus proviendront de la première population, et ils seront ajoutés à la seconde qui compte un grand nombre d'individus de bonnes qualités.
- **La méthode « migration _ best _ worst » :**cette méthode implémente une migration entre deux populations, où certains individus de bonnes qualités migreront de la première vers la seconde qui contienne beaucoup de mauvais individus.

— **La méthode « random _ migration » :** c'est la méthode responsable du dernier type de migration et qui choisis un nombre spécifique d'individus aléatoirement, les extraira de la première population et les ajoutera à la seconde.
— **La méthode « nbr _ indv » :** cette méthode renvoie le nombre d'individus devront migrés vers une population spécifiée.

•**Classe Migration _ step :** cette classe contient deux méthodes qui sont :

— **La méthode « mig _ one _ time » : »** : la méthode exécute l'opération de migration une seule fois dans l'itération spécifiée entre deux populations choisies de manière aléatoire.
— **La méthode « mig _ all » :** pour cette méthode, l'opération de migration peut être exécutée plusieurs fois entre les populations dans une même itération. En effet, toutes les migrations pouvant se produire sont exécutées.

•**Classe Iter _ thread :** cette classe permet l'exécution des étapes de l'algorithme génétique (croisement, mutation, évaluation, sélection) en mode parallèle à l'aide de threads.

•**Classe Resulted _ image :** cette classe contient la méthode qui transformera l'individu solution en une image chiffrée.

•**Classe session _ key :** la classe contient trois méthodes :

— **La méthode « key _ sess » :** cette méthode calcule la clé de session nécessaire pour l'obtention de l'image originale à partir de celle cryptée (déchiffrement).
— **La méthode « write _ key » :** elle permet d'enregistrer la clé de session dans un fichier texte.
— **La méthode « read _ key » :** le procédé permet la lecture de la clé de session à partir du fichier texte où elle a été enregistrée afin d'obtenir l'image originale à partir de celle cryptée.

•**Classe Ga _ app :**cette classe fournit l'exécution de toutes les étapes de l'opération de chiffrement à savoir :

— Lecture de l'image originale introduite pour chiffrement et la transformer en un chromosome initial.
— Division du chromosome initial en sous-chromosomes.
— Création des sous-populations initiales.
— Déroulement des étapes de l'algorithme génétique sur les sous-populations initiales en mode parallèle en exploitant des threads.
— Exécution d'un type de migration là où il faudra.
— Obtention du meilleur individu de chaque sous-population et génération de l'individu final qui représente la meilleure solution.

— Génération de l'image cryptée à partir de l'individu final.
— Obtention et sauvegarde de la clé de session.

• **Classe decryption** :cette classe implémente l'étape de déchiffrement durant laquelle la clé de session sera utilisée afin d'obtenir l'image originale à partir de celle chiffrée.

4.3 Paramètres de chiffrement

Pour appliquer le processus de chiffrement à une image spécifique, certains paramètres doivent être préalablement fixés. Ils concernent les paramètres génétiques, les paramètres de segmentation et les paramètres de migration. Cependant, et vu que notre travail présente une continuité au travail [32], comme c'était précédemment expliqué dans le chapitre de conception, les deux premiers types de paramètres restent inchangés par rapport au travail[32]. Ils seront rappelés ci-dessous. Quand aux paramètres de migration, ils seront étudiés juste après.

4.3.1 paramètres génétiques

Pour les paramètres génétiques qui incluent la probabilité de croisement, la probabilité de mutation, le nombre de générations et la taille des populations, le tableau suivant résume leurs valeurs.

Probabilité du croisement	Probabilité de la mutation	Le nombre de génération	La taille de la population (nombre d'individus)
0.6	0.007	40	10.

TABLE 4.1 – Paramètres génétiques.

4.3.2 paramètres de segmentation

Dans ce travail, une segmentation fixe est celle utilisée. Elle consiste à diviser le chromosome initial en sous-chromosomes de mêmes tailles. Ce type de segmentation a été introduit dans 32 et les tests expérimentaux réalisés quand au taille de segmentation ont conclu sur l'adoption de la taille de la segmentation qui vaut 1000 gènes.

4.3.3 Paramètres de migration

Pour l'étape de migration, deux modes sont distingués :

✓ Le premier consiste à ne faire qu'une migration par itération, ce qui signifie que chaque fois qu'il est nécessaire d'effectuer une migration entre populations (c'est à dire, après un certains nombres d'itérations, ce qui représente un autre paramètre de migration), le type de migration qui convient parmi les trois types précédemment présentés (migration _ best _ worst, migration _ worst _ best, random _ migration) sera appliqué une seule fois entre uniquement deux sous-populations soigneusement choisies.

✓ Pour le second mode, tous les types de migrations possibles seront exécutés entre les sous-populations qui conviennent sur une même itération.

4.4 Résultats expérimentaux

Dans cette section, les résultats des expériences sur une image spécifique (image test) seront présentés. Les tests porteront sur le temps d'exécution et la qualité de l'image résultante (chiffrée) en utilisant les deux modes de migrations (une seule migration par itération ou toutes les migrations possibles sur une même itération) afin de déterminer lequel offrant de meilleurs résultats en termes de qualité et temps d'exécution.

4.4.1 paramètres de tests

Lors de la phase expérimentale, certains paramètres devaient être réglés à savoir : le nombre d'itérations avant migration (avant de réaliser la toute première migration), ce qui signifie que les populations doivent passer par un nombre spécifique d'itérations sans migration, ensuite la migration aura lieu chaque M itérations jusqu'à convergence de l'algorithme. Le tableau suivant présente les détails de ces paramètres.

Nombre d'itération avant la 1 ére migration	migration chaque M itérations	Nombre de migration totale
10	2	15
10	3	10
8	4	8
10	5	6

TABLE 4.2 – Paramètres de tests.

4.4.2 Image test

Le tableau suivant indique les informations relatives à l'image test choisie (image originale) en ce qui concerne son nom et sa dimension, et l'ensemble d'indices qui en résulte à savoir : la taille du chromosome initial résultant de la phase de codage, la taille de segmentation (taille des sous-chromosomes) et le nombre de sous-chromosomes créés suite à l'opération de segmentation et qui représente en même temps le nombre de sous-populations.

Nom Image	Dimension Image (pixels)	Taille du chromosome initial (nbr gènes)	Taille de segmentation (nbr gènes)	Nombre de sous-chromosomes initiaux
monalisa.jpg	35 * 52 (1820 pixels)	5460	1000	6

TABLE 4.3 – Caractéristiques de l'image et indices en résultant.

La figure suivante présente l'image test à utiliser afin de régler expérimentalement les paramètres de l'algorithme.

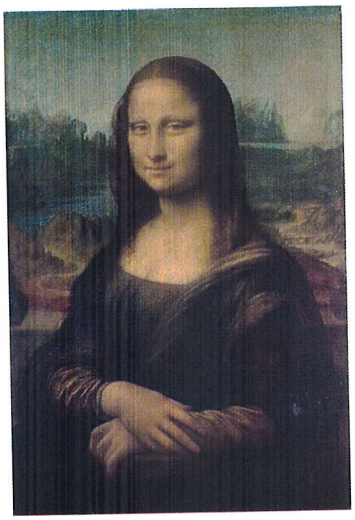


FIGURE 4.1 – Image test (monalisa.jpg).

4.4.3 Tests et résultats

Les sections suivantes présenteront les résultats d'application sur l'image test du processus cryptographique développé suivant les deux modes de migration proposés (une ou toutes les migrations possibles par itération).

4.4.3.1 Premier mode de migration (une seule migration / itération)

a-Une seule migration chaque 2 itérations

Le tableau suivant résume les résultats de dix tests (dix chiffrement séparés de la même image test) concernant le temps d'exécution et l'évaluation de l'image chiffrée retournée dans chacun des cas. Une autre présentation de ces mêmes résultats sera donnée à travers les figures 2 et 3.

Numéro de test	Temps de calcul (ms)	Évaluation
1	951	309304
2	1120	309304
3	864	306118
4	1113	277506
5	1068	283852
6	953	249064
7	999	272688
8	973	280040
9	898	289372
10	924	265456

TABLE 4.4 – Résultats d'une seule migration chaque 2 itérations .

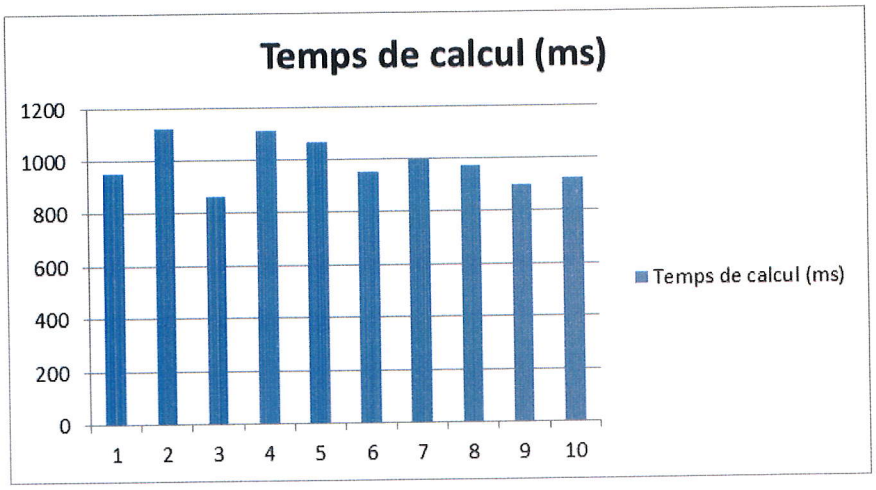


FIGURE 4.2 – histogramme du temps de calcul d'une seule migration chaque 2 itération.

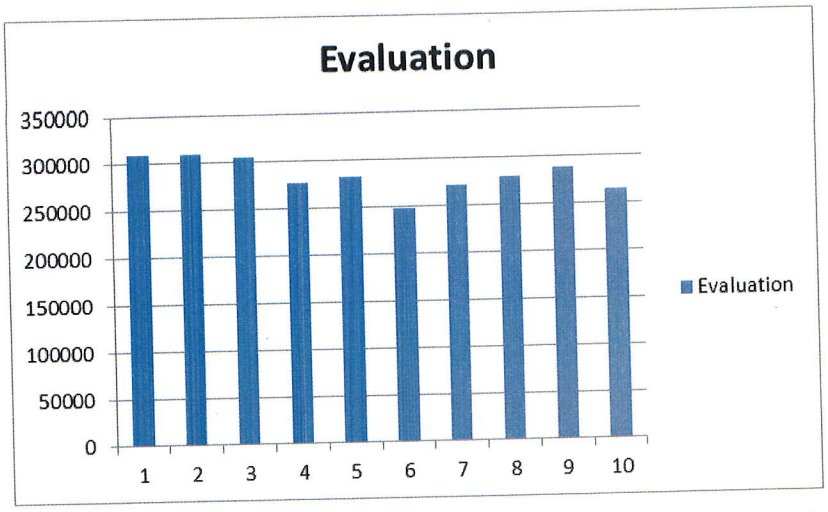


FIGURE 4.3 – histogramme des valeurs d'évaluation d'une seule migration chaque 2 itération.

La figure suivante représente les images résultantes (images chiffrées) des dix tests effectués.

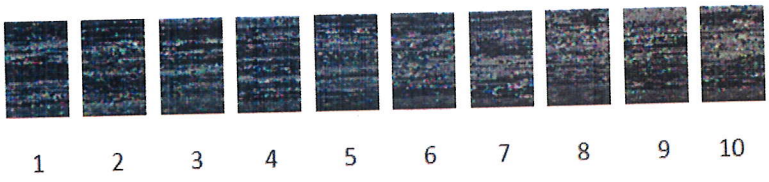


FIGURE 4.4 – Images chiffrées résultantes des dix tests d'une seule migration chaque 2 itérations.

b- Une seule migration chaque 3 itérations

Le tableau suivant en plus des figures 5 et 6, présentent les résultats de dix tests pour ce deuxième cas.

Numéro de test	Temps de calcul (ms)	Évaluation
1	758	289506
2	861	277456
3	808	313530
4	916	300068
5	766	257048
6	1034	296302
7	777	303166
8	772	265856
9	959	290428
10	966	321286

TABLE 4.5 – Résultats d'une seule migration chaque 3 itérations .

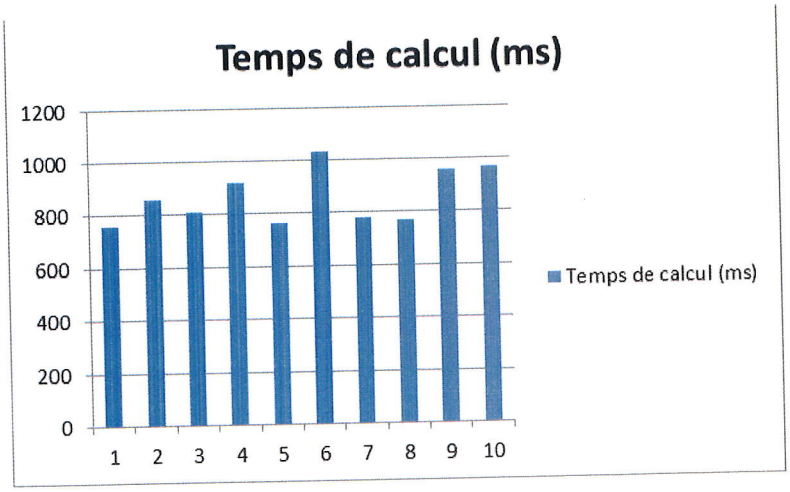


FIGURE 4.5 – histogramme du temps de calcul d'une seule migration chaque 3 itération.

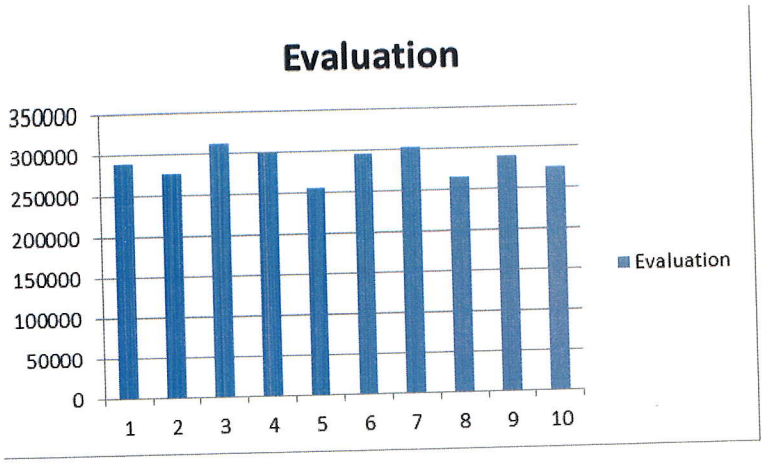


FIGURE 4.6 – histogramme des valeurs d'évaluation d'une seule migration chaque 3 itération.

La figure suivante représente les images chiffrées produites sur les dix tests.

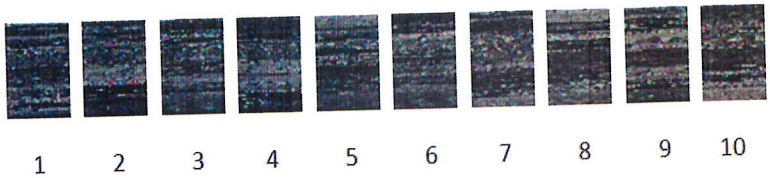


FIGURE 4.7 – Images chiffrées résultantes des dix tests d'une seule migration chaque 3 itérations.

c- Une seule migration chaque 4 itérations

Le tableau suivant et les figures 8 et 9, présentent les résultats de dix tests pour ce quatrième cas.

Numéro de test	Temps de calcul (ms)	Évaluation
1	820	292256
2	798	262638
3	749	246512
4	906	314206
5	812	258170
6	802	306688
7	766	290630
8	902	291904
9	757	273048
10	721	303822

TABLE 4.6 – Résultats d'une seule migration chaque 4 itérations.

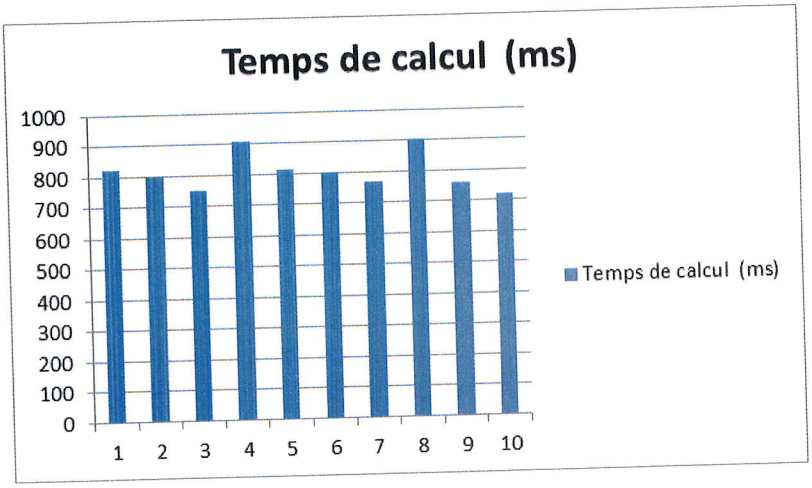


FIGURE 4.8 – histogramme du temps de calcul d'une seule migration chaque 4 itération.

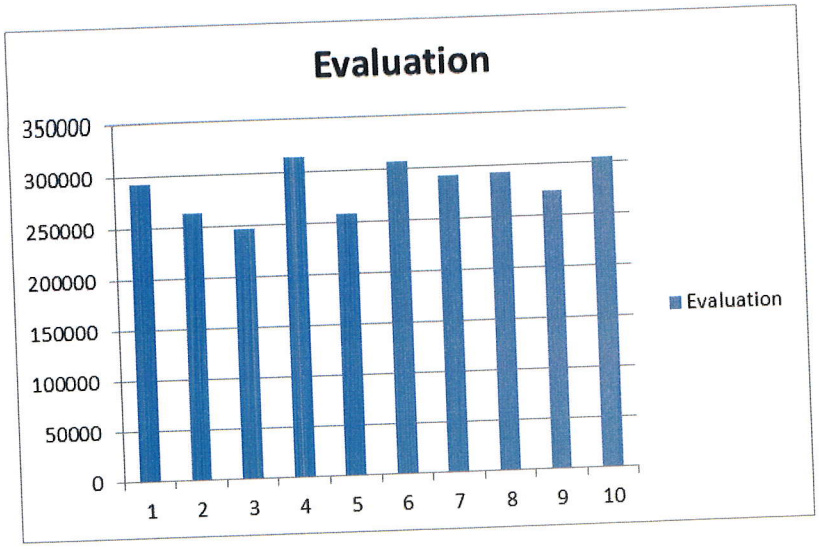


FIGURE 4.9 – histogramme des valeurs d'évaluation d'une seule migration chaque 4 itération.

Les figures suivantes présentent les images résultantes chiffrées des dix fois des tests

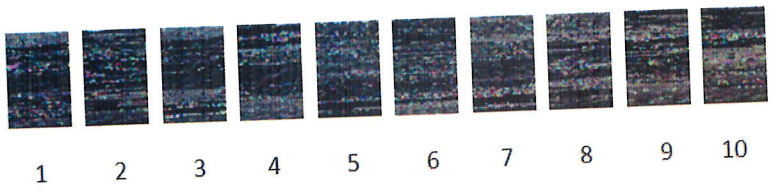


FIGURE 4.10 – Images chiffrées résultantes des dix tests de toutes les migrations possibles chaque 4 itérations.

d- Une seule migration chaque 5 itérations

Le tableau suivant présente les résultats des dix tests sur l'image test choisie pour une seule migration réalisée chaque 5 itérations. De même pour les figures 11 et 12.

Numéro de test	Temps de calcul (ms)	Évaluation
1	730	267678
2	683	299360
3	1054	244880
4	788	246492
5	644	238432
6	725	255130
7	790	241498
8	777	258454
9	737	306220
10	697	294146

TABLE 4.7 – Résultats d'une seule migration chaque 5 itérations .

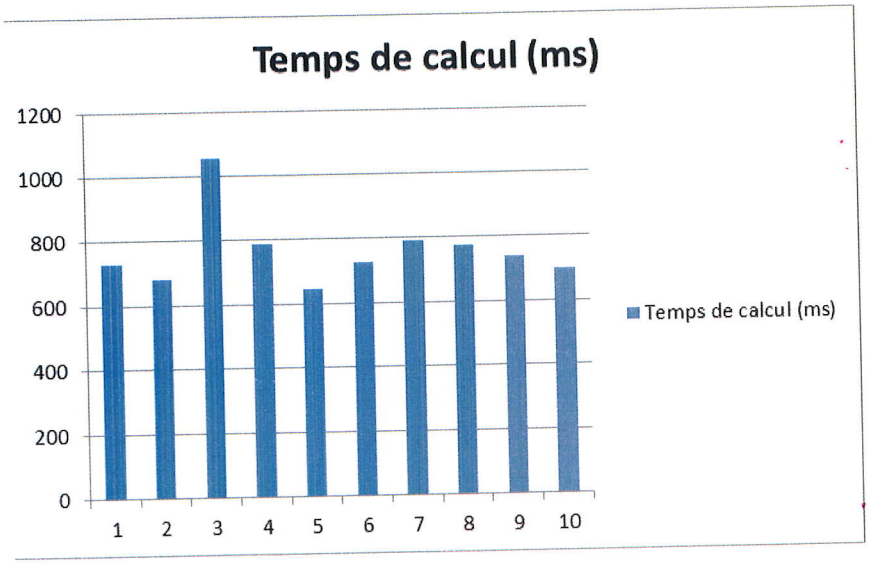


FIGURE 4.11 – histogramme du temps de calcul d'une seule migration chaque 5 itération.

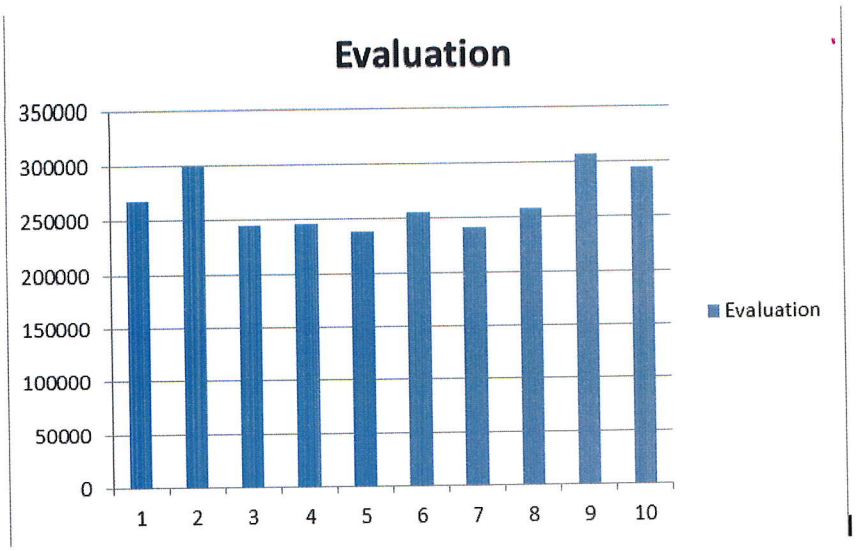


FIGURE 4.12 – histogramme des valeurs d'évaluation d'une seule migration chaque 5 itération.

La figure suivante représente les images résultantes (celles cryptées) des dix fois des tests.

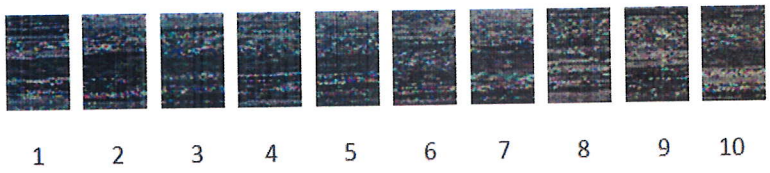


FIGURE 4.13 – Images chiffrées obtenues sur les dix tests d'une seule migration chaque 5 itérations.

Récapitulatif des résultats et interprétation :

Le tableau suivant avec les figures 14 et 15 présentent la moyenne des résultats des dix tests de chacun des quatre cas de d'une seule migration appliquée chaque M itérations

Une seule migration chaque M itérations	Temps de calcul (ms)	L'évaluation
Chaque 2 itérations	986.3	284270.4
Chaque 3 itérations	861.7	291464.6
Chaque 4 itérations	803.3	283987.4
Chaque 5 itérations	762.5	265229

TABLE 4.8 – Moyenne des résultats des quatre cas d'une seule migration.

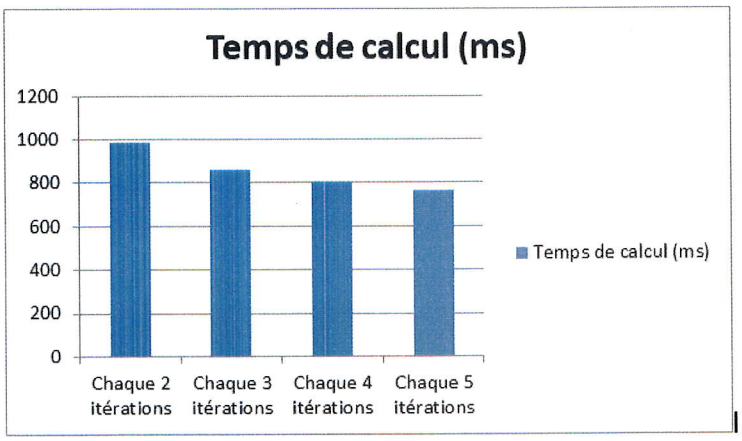


FIGURE 4.14 – Histogramme de moyenne de temps de calcul des quatre cas d'une seule migration.

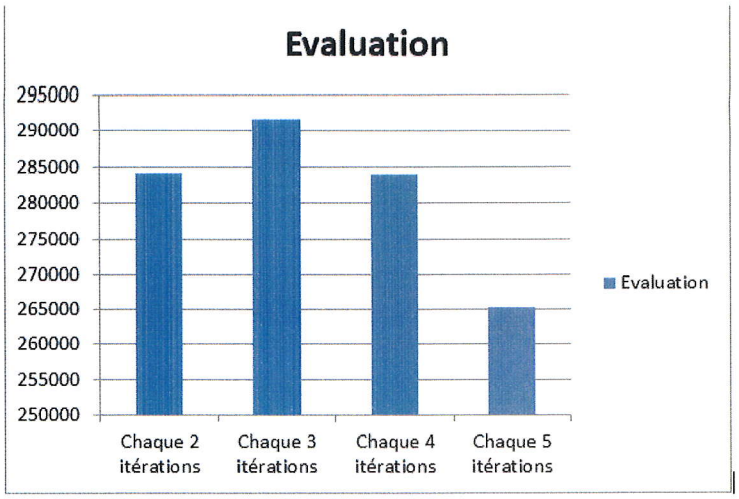


FIGURE 4.15 – Histogramme de moyenne des valeurs d'évaluation des quatre cas d'une seule migration.

D'après les résultats transposés sur la table 8, nous avons constaté le suivant :

- Comparé aux autres cas, les résultats du premier cas qui consistait à réaliser une seule migration chaque 2 itérations, étaient acceptables. En effet, le temps de calcul était le plus lent mais le score d'évaluation était élevé.
- Pour le dernier cas résumant l'adoption d'une seule migration chaque 5 itérations, les résultats étaient acceptables du côté du temps de calcul car il s'agissait du meilleur entre les autres cas. Cependant, et vu que la qualité de l'image chiffrée (sa valeur d'évaluation) est une autre mesure de performance du processus de cryptage ; ceci n'était pas garanti par ce cas de migration car il était le pire parmi les autres cas.
- Les résultats du deuxième et troisième cas (une seule migration chaque 3 et 4 itérations, respectivement) sont proches, mais le deuxième est meilleur.
- En conclusion pour le premier mode de migration (une seule migration), les résultats expérimentaux ont démontrés que le meilleur paramètre était de réaliser une migration chaque 3 itérations donnant lieu à des qualités meilleures en termes de temps de calcul et de qualité d'image chiffrée.

4.4.3.2 Deuxième mode de migration (toutes les migrations possibles / itération)

a- Toutes les migrations possibles chaque 2 itérations

Le tableau suivant résume les résultats des dix tests concernant le temps d'exécution et l'évaluation de l'image chiffrée pour ce cas de migration ainsi que les figures 16 et 17.

Numéro de test	Temps de calcul (ms)	Évaluation
1	1125	288336
2	963	290804
3	909	263814
4	925	301666
5	915	301158
6	1041	334174
7	1041	275762
8	1131	274144
9	943	248750
10	1032	270914

TABLE 4.9 – Résultats des dix tests pour le cas d'application de toutes les migrations possibles chaque 2 itérations.

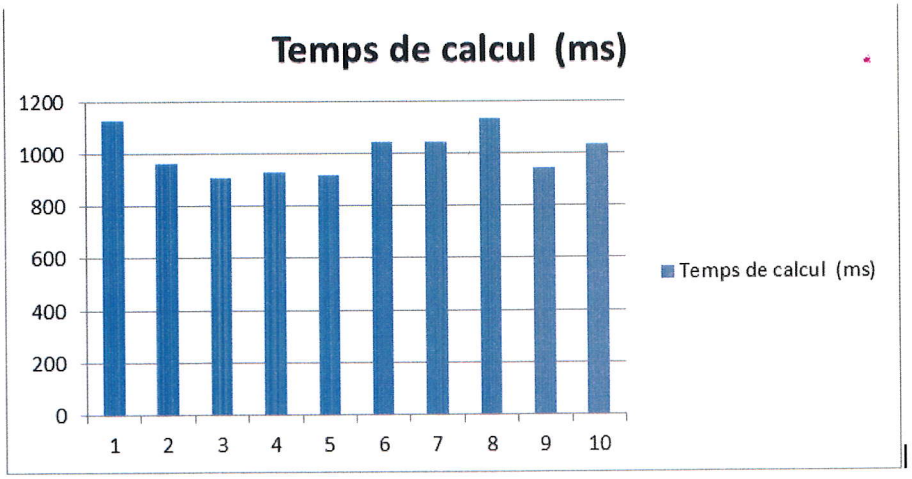


FIGURE 4.16 – histogramme du temps de calcul de toutes les migration possibles chaque 2 itérations.

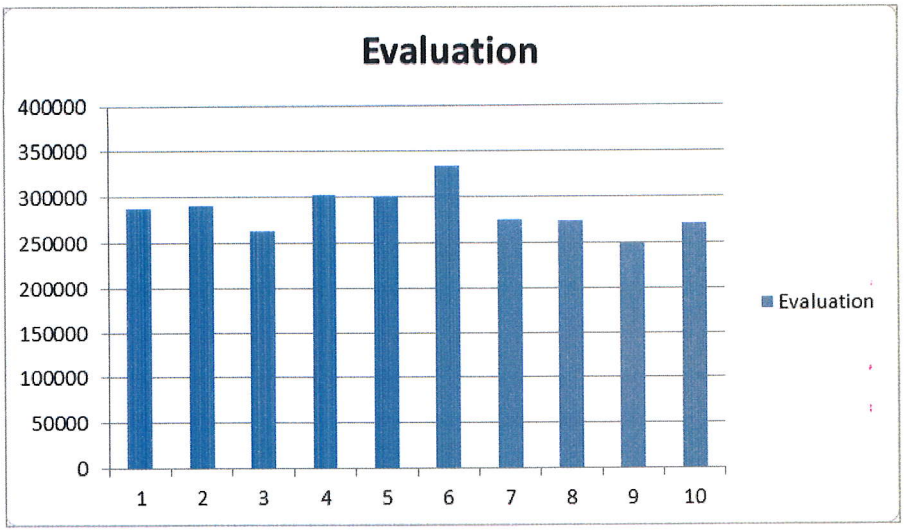


FIGURE 4.17 – histogramme des valeurs d'évaluation de toutes les migrations possibles chaque 2 itérations.

La figure suivante représente les images résultantes (celles cryptées) des dix fois des tests.

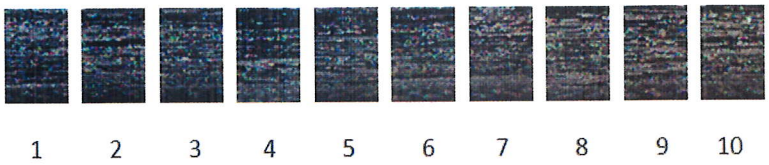


FIGURE 4.18 – Images chiffrées résultantes de toutes les migrations possibles chaque 2 itérations .

b- Toutes les migrations possibles chaque 3 itérations

Le tableau ci-dessous récapitule les résultats des dix tests pour ce cas de migration. De même pour les figures 19 et 20.

Numéro de test	Temps de calcul (ms)	Évaluation
1	800	265084
2	923	304408
3	825	291680
4	829	305060
5	817	305424
6	813	268392
7	906	292954
8	914	275856
9	969	280234
10	845	285142

TABLE 4.10 – Résultats des dix tests de toutes les migrations possibles chaque 3 itérations.

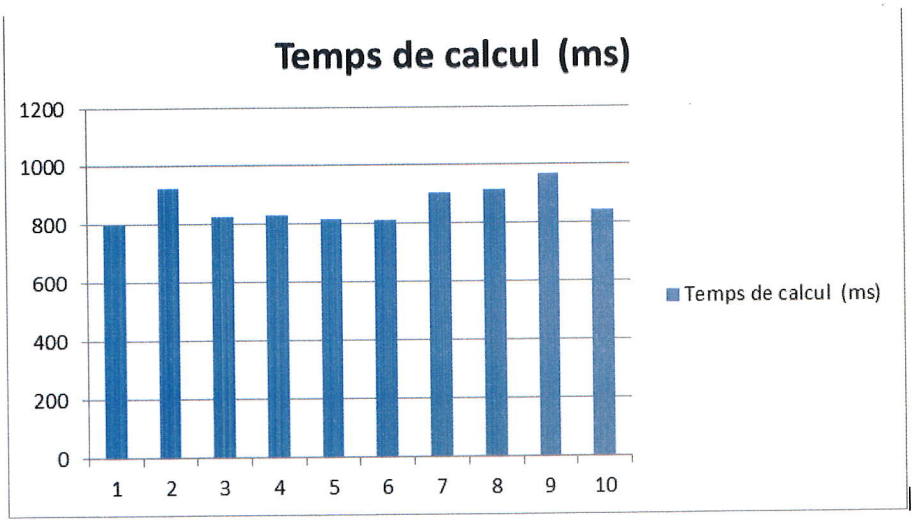


FIGURE 4.19 – histogramme du temps de calcul des dix tests pour le cas de toutes les migrations possibles chaque 3 itérations.

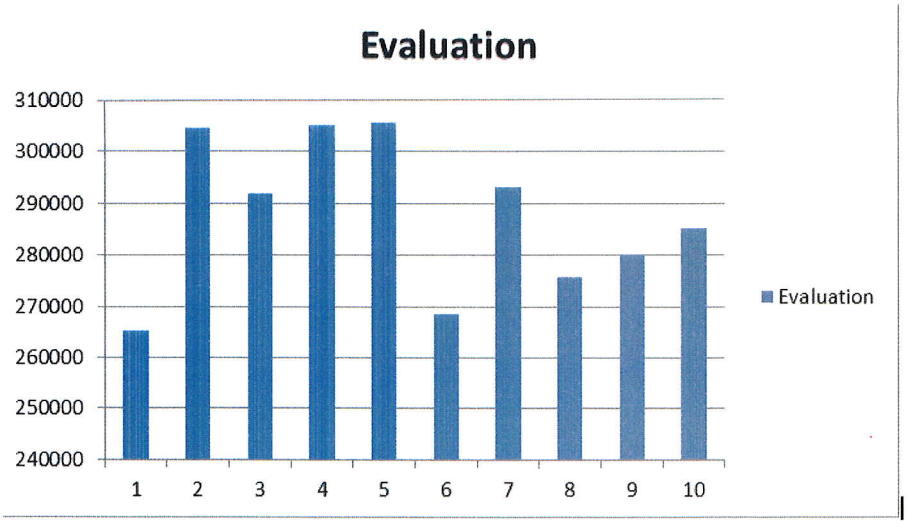


FIGURE 4.20 – histogramme des valeurs d'évaluation des dix tests de toutes les migrations possibles chaque 3 itérations.

La figure ci-dessous présente les images chiffrées résultantes sur les dix tests correspondant à ce cas de migration.

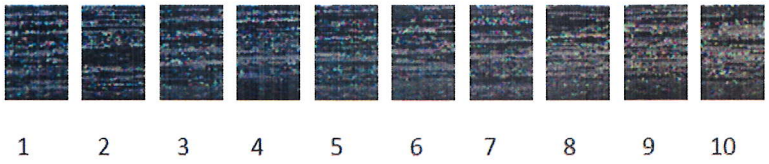


FIGURE 4.21 – Images chiffrées résultantes des dix tests pour le cas d'application de toutes les migrations possibles chaque 3 itérations .

c- Toutes les migrations possibles chaque 4 itérations

Le tableau suivant présente les résultats de dix tests en termes de temps d'exécution et de valeurs d'évaluation de l'image résultante pour ce cas de migration. De même pour les figures 22 et 23.

Numéro de test	Temps de calcul (ms)	Évaluation
1	774	261384
2	781	284086
3	761	243600
4	838	317298
5	828	306814
6	765	296012
7	735	269452
8	875	279704
9	853	305738
10	926	243066

TABLE 4.11 – Résultats des dix tests de toutes les migrations possibles chaque 4 itérations.

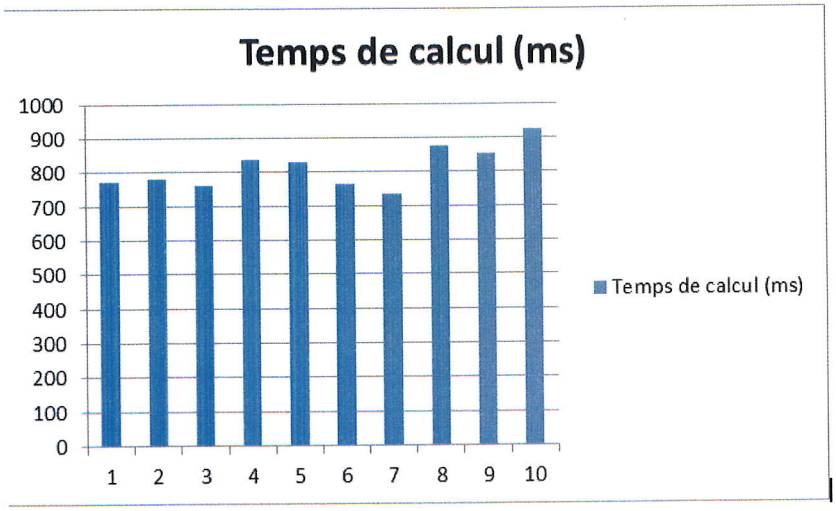


FIGURE 4.22 – histogramme du temps de calcul des dix tests de toutes les migrations possibles chaque 4 itérations.

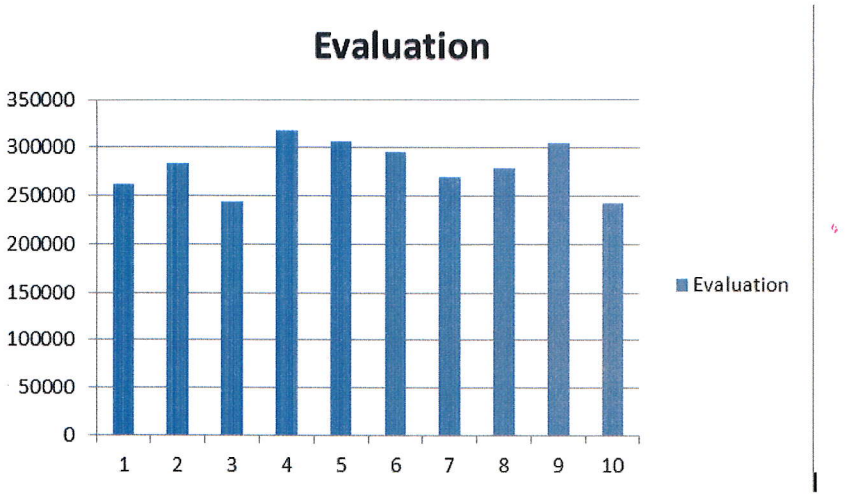


FIGURE 4.23 – histogramme des valeurs d'évaluation des dix tests de toutes les migrations possibles chaque 4 itérations.

La figure ci-dessous présente les images chiffrées des dix fois des tests.

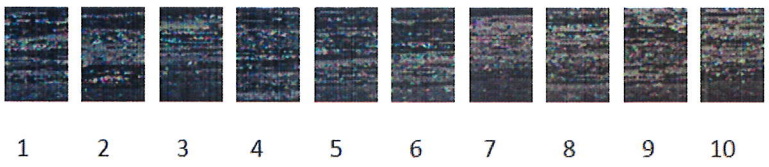


FIGURE 4.24 – Images chiffrées résultantes des dix tests de toutes les migrations possibles chaque 4 itérations.

d- Toutes les migrations possibles chaque 5 itérations

Le tableau suivant en plus des figures 25 et 26, résument les résultats des dix tests d'application de ce cas de migration sur l'image test choisie.

Numéro de test	Temps de calcul (ms)	Évaluation
1	869	275216
2	792	312290
3	680	250804
4	734	290092
5	843	312004
6	751	307852
7	775	311538
8	703	298708
9	784	276024
10	746	289720

TABLE 4.12 – Résultats des dix tests d'application de toutes les migrations possibles chaque 5 itérations.

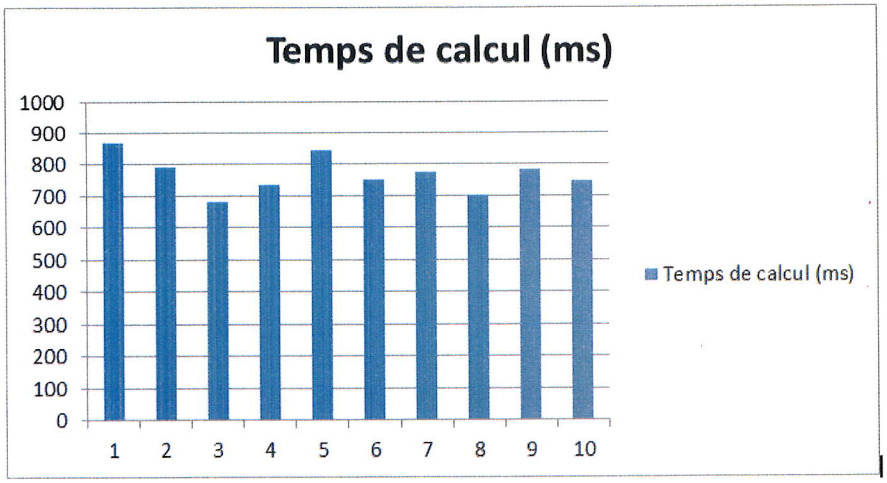


FIGURE 4.25 – histogramme du temps de calcul des dix tests de toutes les migrations possibles chaque 5 itérations.

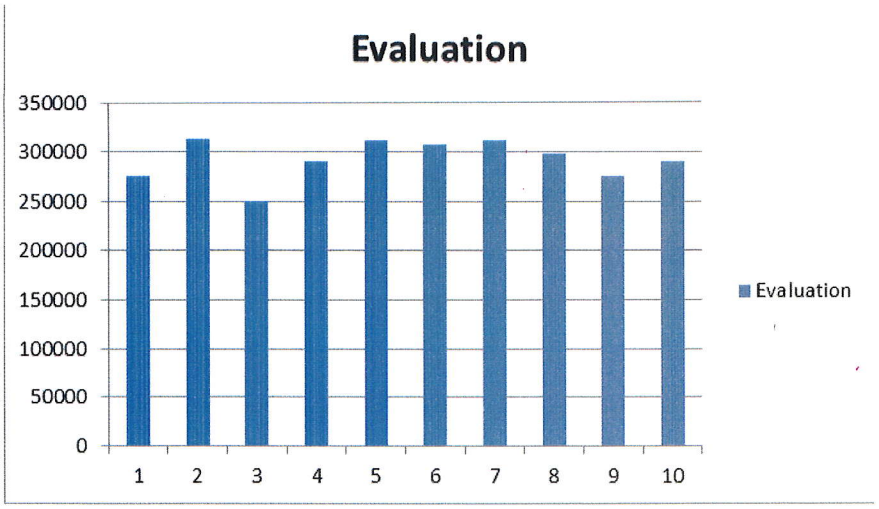


FIGURE 4.26 – histogramme des valeurs d'évaluation obtenues sur les dix tests de toutes les migration possibles chaque 5 itérations

La figure ci-dessous présente les images chiffrées obtenues sur les dix tests précédemment présentés.

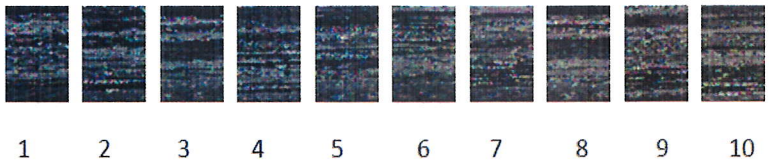


FIGURE 4.27 – Images chiffrées obtenues sur les dix tests de toutes les migrations possibles chaque 5 itérations.

Récapitulatif des résultats et interprétation : Le tableau suivant avec les figures 28 et 29 présentent la moyenne des résultats obtenus sur les dix tests des différents cas du deuxième mode de migration.

Une seule migration chaque M itérations	Temps de calcul (ms)	L'évaluation
Chaque 2 itérations	1002.5	284952.2
Chaque 3 itérations	864.1	287423.4
Chaque 4 itérations	813.6	280715.4
Chaque 5 itérations	767.7	292424.8

TABLE 4.13 – Moyenne des résultats des dix tests d'application de toutes les migrations possibles sur une itération donnée.

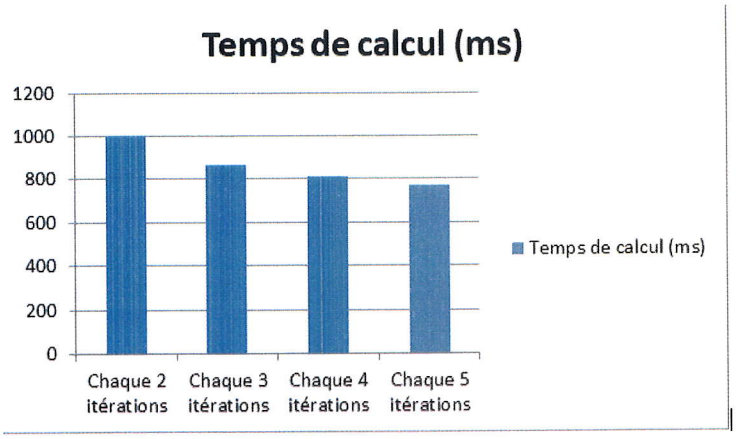


FIGURE 4.28 – Histogramme de moyenne de temps de calcul des quatre cas de toutes les migrations possibles.

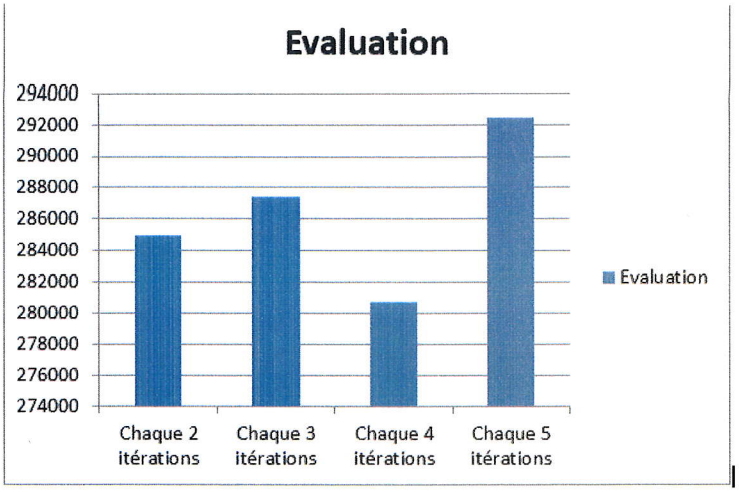


FIGURE 4.29 – Histogramme de moyenne des valeurs d'évaluation des quatre cas de toutes les migrations possibles.

D'après les résultats obtenus suivant les différents cas de migration pour ce deuxième mode (voir table 13), les points suivants ont été tirés :

- le temps de calcul le plus lent a été obtenu dans le premier cas (toutes les migrations possibles chaque 2 itérations), cependant, le score d'évaluation était élevé.
- Le score d'évaluation le plus bas (mauvais) a été obtenu dans le troisième cas (toutes les migrations possibles chaque 4 itérations) avec un temps de calcul qui peut être considérer acceptable.
- le deuxième cas (toutes les migrations possibles chaque 3 itérations) peut être considéré comme acceptable de côtés de score d'évaluation mais les résultats de temps de calcul n'était pas très satisfaisant par rapport aux autres cas. Cependant, le meilleur est le quatrième cas (toutes les migrations possibles chaque 5 itérations) avec le temps de calcul le plus rapide et le score d'évaluation le plus élevé. En conclusion des deux modes de migration proposés, le meilleur paramétrage pour le premier est de réaliser une seule migration chaque 3 itérations et de réaliser toutes les migrations possibles chaque 5 itérations pour le deuxième mode de migration. Et comme le deuxième mode de migration est meilleur que le premier, donc nous suggérons d'adopter ce mode pour notre application.

  Donc, pour le paramétrage relatif à la migration, nous opterons pour la réalisation de toutes les migrations possibles, sur une même itération, chaque 5 itérations.

4.4.4 Variation du paramètre relatif au nombre d'individus

Une fois le meilleur paramétrage de migration a été fixé, nous testerons, ici, l'impact d'une augmentation du nombre d'individus dans une population sur l'efficacité de l'algorithme

proposé. Il est clair qu'une telle chose augmentera le temps de calcul, mais on essayera de conclure sur l'impact d'une telle modification sur la qualité de l'image chiffrée obtenue dans ce cas. Pour cela, des tests ont été réalisés à cinq reprises pour un nombre d'individus égale à 20, 30, 40 et 50.
Les tableaux suivants présentent les résultats des tests en tenant compte du temps d'exécution et du score d'évaluation. Les tests ont été effectués sur la même image que celle utilisée pour les tests précédents (figure 1).

$$a - Nombred'individus = 20$$

Numéro de test	Temps de calcul (ms)	Évaluation
1	946	241014
2	1000	235908
3	825	271106
4	1018	277206
5	838	301722

TABLE 4.14 – Résultats des tests pour un nombre d'individus = 20.

$$b - Nombred'individus = 30$$

Numéro de test	Temps de calcul (ms)	Évaluation
1	1271	247990
2	1329	239170
3	1283	234424
4	1169	235396
5	1187	265634

TABLE 4.15 – Résultats des tests pour un nombre d'individus =30.

$$c - Nombred'individus = 40$$

Numéro de test	Temps de calcul (ms)	Évaluation
1	1684	236032
2	1639	232408
3	1377	236174
4	1372	233754
5	1493	289382

TABLE 4.16 – Résultats des tests pour un nombre d'individus =40.

$$d - Nombred'individus = 50$$

Numéro de test	Temps de calcul (ms)	Évaluation
1	1973	300672
2	1519	240494
3	1819	239366
4	1749	235258
5	1700	236758

TABLE 4.17 – Résultats des tests pour un nombre d'individus = 50.

Récapitulatif des résultats et interprétation :

La table suivante avec les figures 30 et 31 présente la moyenne des 5 tests résumés ci-dessus pour les différentes valeurs de nombre d'individus testées.

Nombre d'individus	Temps de calcul (ms)	Évaluation
20	925.4	265391.2
30	1247.8	244522.8
40	1513	245550
50	1752	250509.6

TABLE 4.18 – Moyenne des résultats pour les différentes valeurs de nombre d'individus.

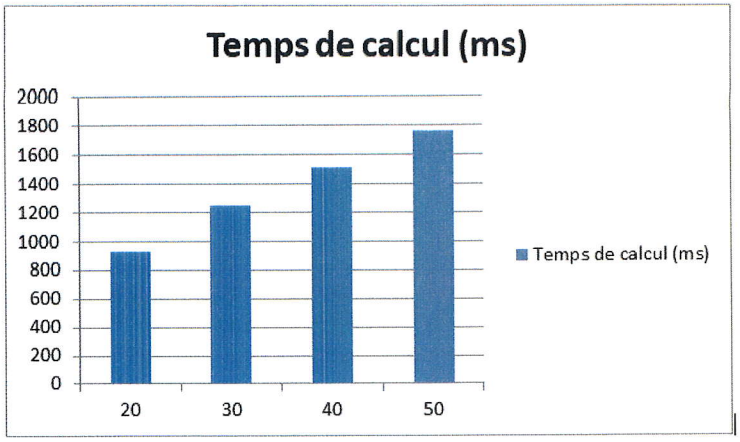


FIGURE 4.30 – Histogramme de moyenne de temps de calcul pour les différentes valeurs de nombre d'individus.

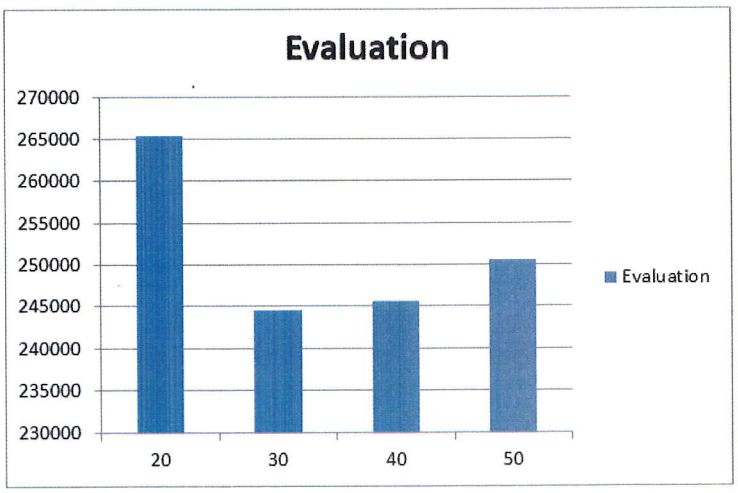
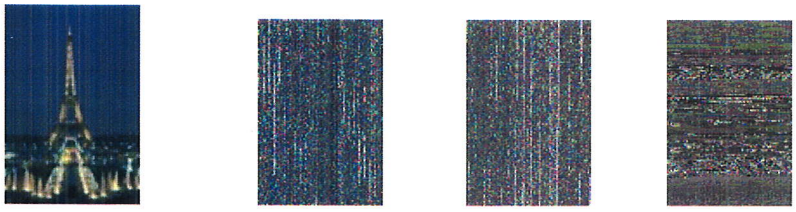


FIGURE 4.31 – Histogramme de moyenne des valeures d'évaluation pour les différentes valeurs de nombre d'individus.

Les résultats des tests résumés à travers la table 18, montrent que l'augmentation en nombre d'individus dans une population (c'est à dire , augmenter la taille de populations) n'a eu aucun impact sur la qualité de l'image cryptée. Au contraire, soit qu'elle reste inchangée ou elle baisse. Donc, compte tenu de ce constat accompagné d'une augmentation en temps de calcul, nous suggérons de garder un nombre d'individus égal à 10.

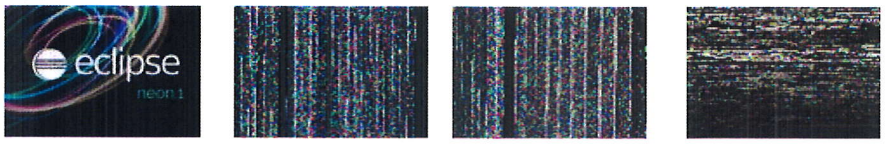
4.5 Résultats expérimentaux et comparaison

Après avoir décrit en détail le paramétrage adopté, nous rapportons, dans cette section, les résultats d'application de notre algorithme proposé, appelé PFSPosMig pour Parallel Fixed Segmentation Position based with migration, en comparaison avec ceux obtenus par l'algorithme de base, PosESecL2,et les deux algorithmes PFSPos et PVSPos proposés par [32] sur des images de différentes tailles.



(a) (b) (c) (d)

FIGURE 4.32 – image test Effel (a) image original, (b) image chiffrée par PFSPos , (c) image chiffrée par PVSPos, (d) image chiffrée par PFSPosMig.



(b) (b) (c) (d)

FIGURE 4.33 – image test Eclipse (a) image original, (b) image chiffrée par PFSPos , (c) image chiffrée par PVSPos, (d) image chiffrée par PFSPosMig.

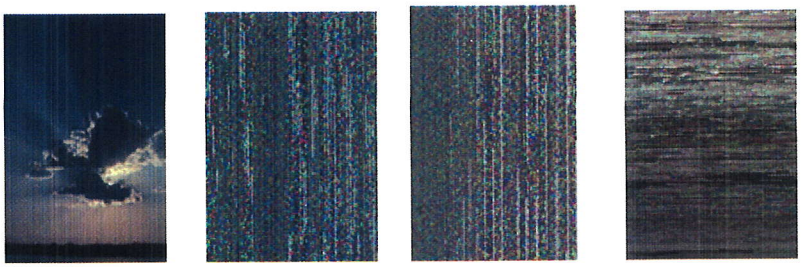


FIGURE 4.34 – image test sky (a) image original, (b) image chiffrée par PFSPos , (c) image chiffrée par PVSPos, (d) image chiffrée par PFSPosMig.

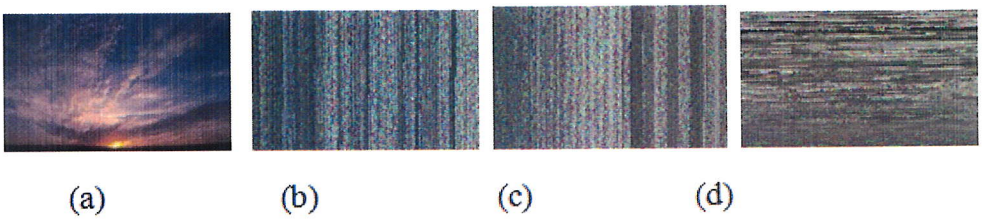
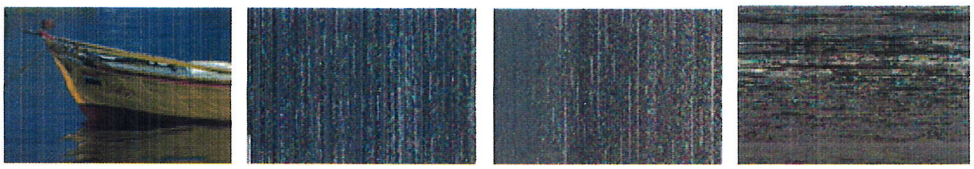
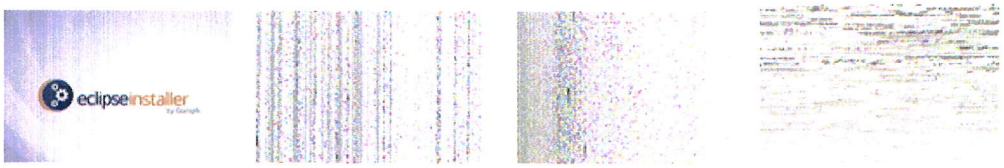


FIGURE 4.35 – image test Eclipse (a) image original, (b) image chiffrée par PFSPos , (c) image chiffrée par PVSPos, (d) image chiffrée par PFSPosMig.



(a) (b) (c) (d)

FIGURE 4.36 – image test Boat (a) image original, (b) image chiffrée par PFSPos , (c) image chiffrée par PVSPos, (d) image chiffrée par PFSPosMig.



(a) (b) (c) (d)

FIGURE 4.37 – image test Eclipse2 (a) image original, (b) image chiffrée par PFSPos , (c) image chiffrée par PVSPos, (d) image chiffrée par PFSPosMig.

Après l'adoption des valeurs choisies pour le paramétrage nous reportons en ce qui suit les résultats obtenus du chiffrement et déchiffrement des images : Effel, Eclipse, sky , Sky2, Boat et Eclipse2.
Le tableau suivant donne des idées sur les tailles de la clé de session générée par l'algorithme PFSPosMig pour chacune des images de test ainsi que le temps de chiffrement, de déchiffrement et les valeurs d'évaluation des images chiffrées
obtenues.

Images	Taille image(pixels)	Taille clé (bits)	Temps de chiffrement(s)	Temps de déchiffrement(s)	Évaluation
Effel	72 * 108	349920	6.56	0.2386	902144,66
Eclipse	113 * 76	386460	6.9112	0.1	1498406
sky	100 * 150	720000	19.1596	0.1168	2398534
Sky2	180 * 108	933120	29.078	0.426	2812598.6
Boat	216 * 143	1575288	58.3562	0.1936	4273242.8
Eclipse2	226 * 151	1728900	99.88	0.145	2444759.3

TABLE 4.19 – Résultats de manipulation des six images test par l'algorithme PFSPosMig.

D'après les résultats présentés dans le tableau 19 , il est claire que la taille de la clé augmente proportionnellement avec l'augmentation de la taille de l'image à chiffrer. La même remarque peut être faite sur le temps de chiffrement et le temps de déchiffrement. Cependant, le temps de déchiffrement est très court comparé au temps de chiffrement. Par exemple, l'image test sky ayant une taille égale à (100 * 150) pixels consomme 19,1596 secondes pour être chiffrer mais sauf 0,1168 seconde pour être déchiffrer. De même, l'image test Eclipse2 ayant une taille égale à (226 * 151) pixels, a pris 99,88 secondes pour être chiffrer mais l'image déchiffrée a résulté après seulement 0,145 seconde. Les deux tableaux suivants donnent des idées sur les tailles de la clé de session générée par les deux algorithmes PFSPos et PVSPos pour chacune des six images de test ainsi que le temps de chiffrement, temps de déchiffrement et valeurs d'évaluation des images chiffrées résultantes.

Images	Taille image(pixels)	Taille clé (bits)	Temps de chiffrement(s)	Temps de déchiffrement(s)	Évaluation
Effel	72 * 108	349920	5.917	5.307	960183
Eclipse	113 * 76	386460	6.7822	6.1284	1669161
sky	100 * 150	720000	9.666	21.1176	2218899.4
Sky2	180 * 108	933120	12.57	35.287	2569187
Boat	216 * 143	1575288	16.752	114.8902	4237298.8
Eclipse2	226 * 151	1728900	25.737	150.360	1795460.33

TABLE 4.20 – Résultats de manipulation des six images test par l'algorithme PFSPos .

Images	Taille image(pixels)	Taille clé (bits)	Temps de chiffrement(s)	Temps de déchiffrement(s)	Évaluation
Effel	72 * 108	349920	6.59	5.061	955937.33
Eclipse	113 * 76	386460	8.6144	6.2168	1663069.2
sky	100 * 150	720000	15.9204	18.324	2239470.6
Sky2	180 * 108	933120	23.751	35.1876	2570329.3
Boat	216 * 143	1575288	38.2178	100.6336	4217156 .2
Eclipse2	226 * 151	1728900	72.393	128.268	1788250.33

TABLE 4.21 – Résultats de manipulation des six images test par l'algorithme PVSPos.

La même remarque faite sur l'augmentation de taille de clé, de temps de chiffrement et de temps de déchiffrement proportionnellement avec l'augmentation des tailles d'images soumises au chiffrement par PFSPosMig, reste valable pour l'application des algorithmes PFSPos et PVSPos. Par exemple et d'après les résultats d'application des algorithmes PFSPos et PVSPos, transposés dans les tableaux 20 et 21 , l'image Effel de taille égale à (72 * 108) pixels, a pris 5,917 secondes pour temps de chiffrement et 5,307 secondes pour temps de déchiffrement par PFSPos et 6,59 secondes pour temps de chiffrement et 5,061 secondes pour temps de déchiffrement par PVSPos. Toutefois, l'image Boat ayant une taille égale à (216 * 143) pixels qui est une taille plus grande de la taille de l'image Effel, a pris 16,752 secondes pour temps de chiffrement et 114,8902 secondes pour temps de déchiffrement par PFSPos et 38,2178 secondes pour temps de chiffrement et 100,6336 secondes pour temps de déchiffrement par PVSPos.

En comparant, maintenant, notre algorithme PFSPosMig, avec les deux versions parallèles de PosSEsecL2 qui sont PFSPos et PVSPos, les points suivants ont été constatés :

- ✓ En terme de temps de déchiffrement, PFSPosMig est l'algorithme le plus rapide, suivi de PVSPos et le PFSPos est le plus lent.
- ✓ Toutefois et pour l'opération de chiffrement, PFSPos est l'algorithme le plus rapide suivi de PVSPos, et le PFSPosMig est le plus lent.
- ✓ Pour l'évaluation de la qualité des images chiffrées, PFSPosMig est le plus sécuritaire (car et comme la fonction d'évaluation adoptée par les trois algorithme mesure la différence entre l'image originale et l'image chiffrée correspondante, donc, un grand score d'évaluation traduit un grand pouvoir confusionnel). En effet, les scores d'évaluation par PFSPosMig des images chiffrées correspondantes aux images présentées ci-dessus, sont les plus grands à part ceux correspondant aux images "effet" et "eclipse" qui sont des images de petites tailles par rapport aux autres images. Toutefois PFSPos et PVSPos sont presque de même niveau sécuritaire.

Ces notes tirées expérimentalement reviennent au fait que l'adoption du modèle en îlots avec migrations (utilisé par PFSPosMig) consomme plus de temps que l'application du même modèle mais sans migration (utilisé par PFSPos et PVSPos). Cela est du au temps d'échange d'information entre les ilots (modules parallèles) surtout pour le cas du mode de migration adopté par PFSPosMig qui consiste à réaliser toutes les migrations possibles sur les mêmes itérations. Mais d'un autre coté, ce dernier fait (réaliser toutes les migrations possibles sur les mêmes itérations dans le modèles en îlots avec migrations) a doté le PFSPosMig d'un pouvoir confusionnel plus important que celui des deux autres algorithmes exploitant le modèle en îlots sans migration.
Maintenant, pour les deux algorithmes PFSPos et PVSPos exploitant tout les deux un même modèle, c'est le mode de segmentation qui fait la différence du temps de convergence des deux algorithme. En effet, une segmentation variable consomme plus de temps qu'une segmentation fixe et une manipulation de sous-populations toutes de mêmes tailles consomme moins de temps qu'une manipulation de sous-populations de différentes tailles augmentées d'une sous-population à une autre d'un certain pas. C'est pourquoi le PFSPos était plus rapide que PVSPos.
En comparant, maintenant, les quatre algorithmes génétiques de chiffrement basé positions (PosESecL2, PFSPos, PVSPos et PFSPosMig), il est clair que la version séquentielle (PosESecL2) est la plus lente du fait que le codage manipulé dépend étroitement de la taille de l'image à chiffrer, tandis qu'il est indépendant de cela dans les trois versions parallèles mais le temps de calcul dépend, plutôt, des paramètres de segmentation (taille et pas de segmentation pour PFSPos et PVSPos, respectivement) et de l'opération de migration pour PFSPosMig.

4.6 Interfaces (fenêtres) de l'application

Le but de notre application est de chiffrer une image à l'aide d'un algorithme génétique parallèle. A travers notre conception de fenêtres répondant aux différents cotés du travail et ses différentes fonctionnalités), nous cherchions à ce que l'application soit d'un usage simple. Dans cette section, nous décrivons les différentes fenêtres constituant l'application.

4.6.1 la fenêtre d'accueil

Lorsque l'utilisateur lance l'application, la première fenêtre qui s'affiche est la fenêtre d'accueil (figure 38). Cette fenêtre lui donne la possibilité d'accéder aux principales fonctionnalités de l'application en cliquant sur le bouton « Entrer », ou de quitter l'application en utilisant le bouton « Quitter ».

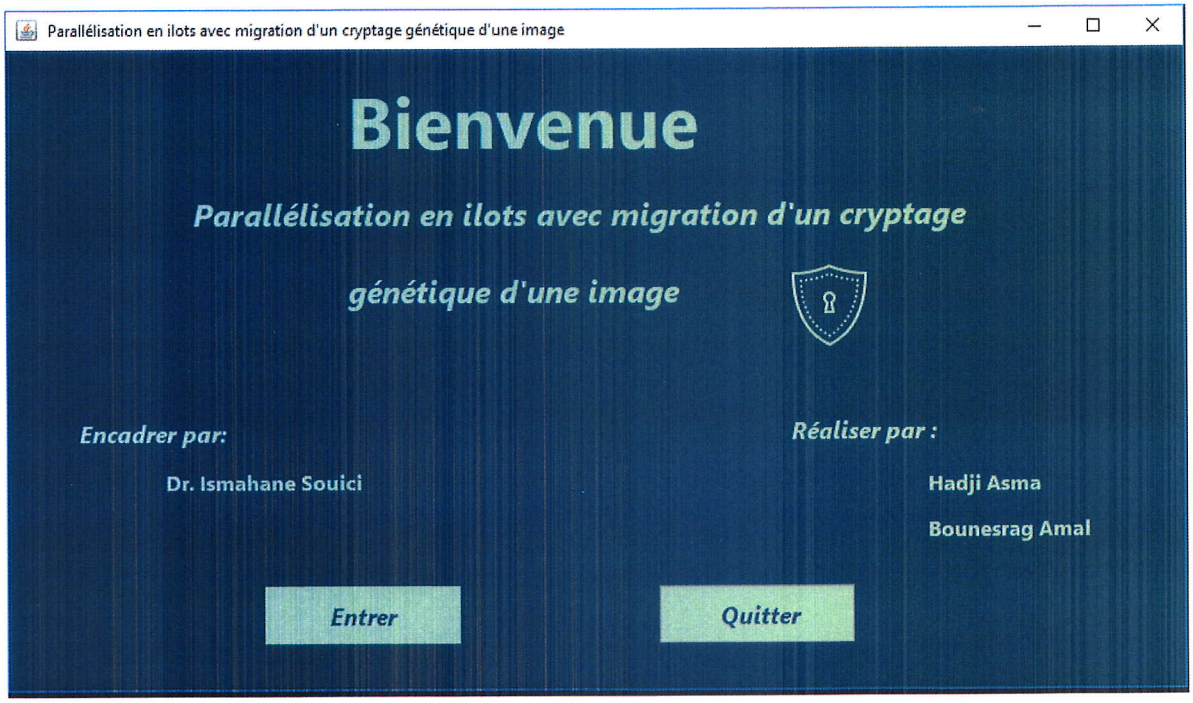


FIGURE 4.38 – la fenêtre d'accueil .

4.6.2 La fenêtre du chiffrement d'une image

Cette fenêtre donne à l'utilisateur la possibilité de chiffrer une image soit avec une seule migration ou avec toutes les migrations possibles, de déchiffrer l'image chiffrée et d'accéder aux informations supplémentaires sur l'application. (figure 39)

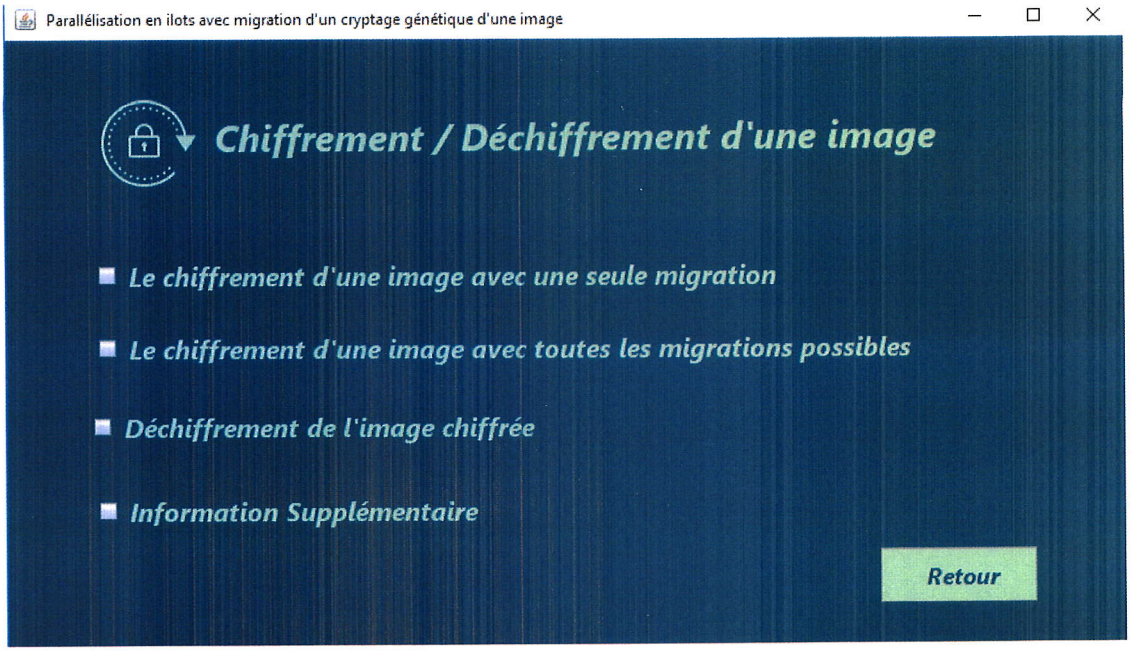


FIGURE 4.39 – la fenêtre du chiffrement / déchiffrement d'une image .

4.6.2.1 Chiffrement d'une image avec une seule migration

Cette option offre un chiffrement d'une image en réalisant une seule migration par itération en respectant le paramétrage étudié ci-dessus. La fenêtre (figure 40) contient quatre boutons permettant à l'utilisateur de réaliser les opérations suivantes :

- télécharger une image en utilisant le bouton « Télécharger ».
- La possibilité de modifier les paramètres en utilisant le bouton « Modifier les paramètres ».
- Exécution de l'opération de chiffrement de l'image à l'aide du bouton « chiffrer ».
- Retour à la fenêtre de « Chiffrement / Déchiffrement d'une image » en utilisant le bouton « Retour ».
- Il existe également un champ à travers lequel l'utilisateur est inviter à proposer un nom pour l'image chiffrée avant d'exécuter l'opération de chiffrement.

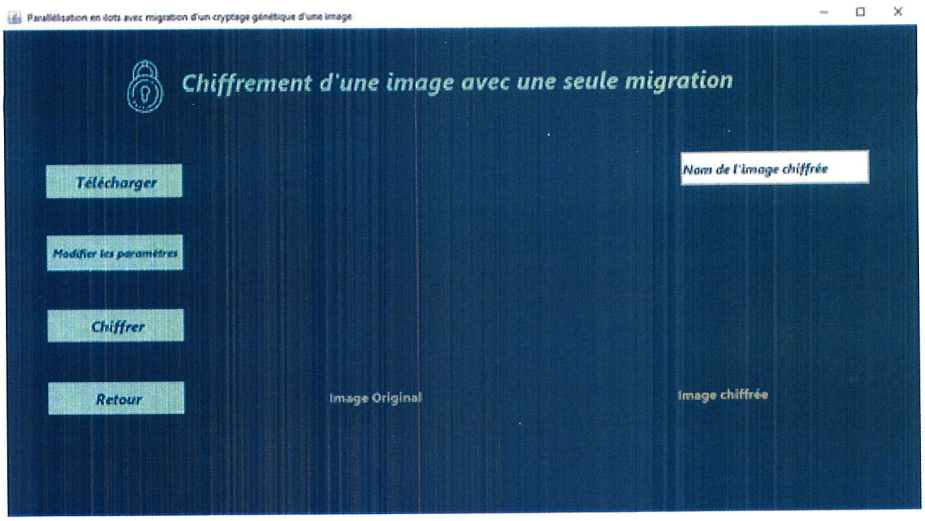


FIGURE 4.40 – La fenêtre de chiffrement d'une image avec une seule migration.

Après avoir charger une image originale et proposer un nom pour l'image chiffrée (voir exemple sur la figure 41), l'opération de chiffrement peut être lancer en cliquant sur le bouton « Chiffrer » et les résultats seront, ensuite, affichés (l'image chiffrée, le temps de calcul et l'évaluation de l'image chiffrée obtenue (voir exemple sur la figure 42)).

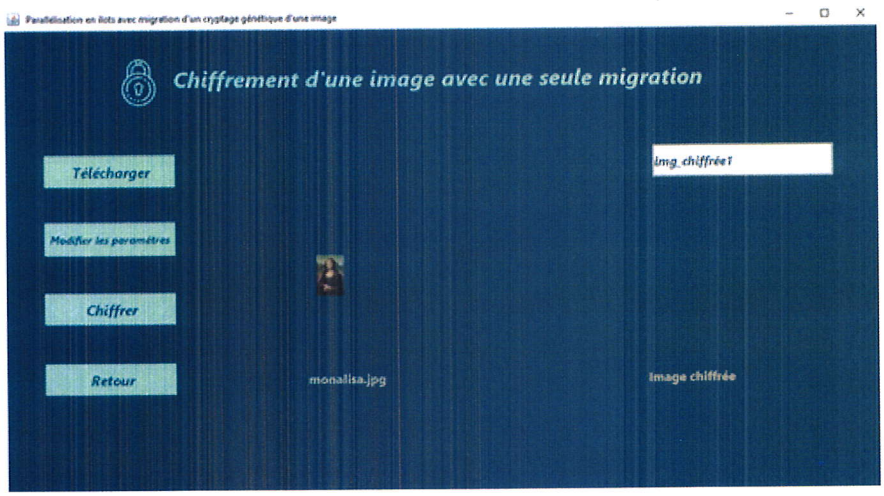


FIGURE 4.41 – Exemple de chargement d'une image originale et proposition d'un nom pour l'image chiffrée à calculer.

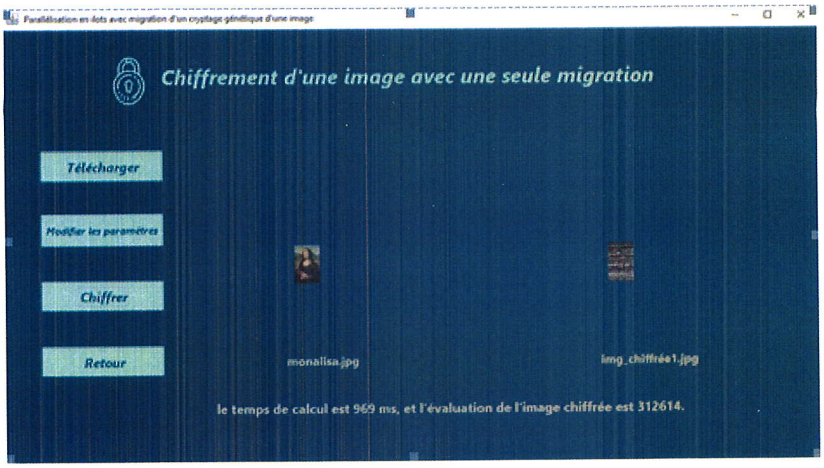


FIGURE 4.42 – Résultats de chiffrement.

Supposons qu'un utilisateur tente d'effectuer une opération de chiffrement sans télécharger une image originale en cliquant sur le bouton « chiffrer », un message d'erreur s'affichera lui demandant de télécharger une image originale tout d'abord (voir figure 43).

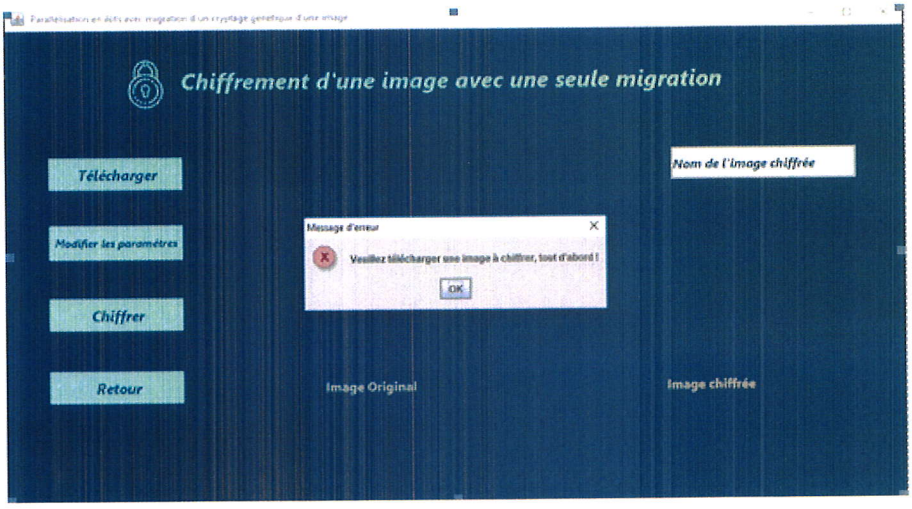


FIGURE 4.43 – Message indiquant la nécessité de télécharger une image originale avant de tenter un chiffrement.

La même chose se produira si l'utilisateur tente de chiffrer sans proposer un nom pour l'image chiffrée à produire (voir figure 44).

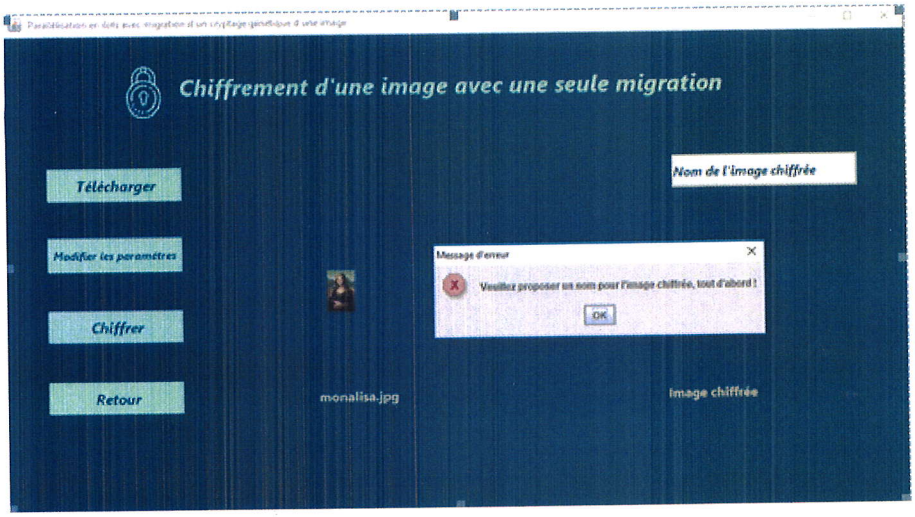


FIGURE 4.44 – Message indiquant la nécessité de proposer un nom pour l'image chiffrée avant de tenter un chiffrement.

4.6.2.2 Chiffrement d'une image avec toutes les migrations possibles

Les composants de cette fenêtre sont exactement les mêmes que ceux de la fenêtre de chiffrement avec une seule migration, c'est le détail de l'opération de chiffrement qui diffère en faisant référence à un chiffrement réalisant toutes les migrations possibles sur les mêmes itérations.

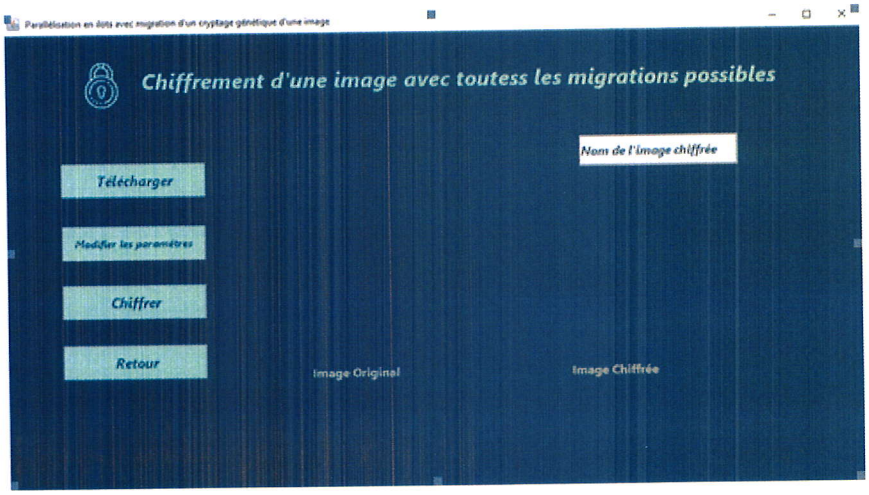


FIGURE 4.45 – Fenêtre de chiffrement d'une image avec toutes les migrations possibles.

4.6.2.3 Fenêtre de modification de paramètres

Lorsque l'utilisateur clique sur le bouton « Modifier paramètres », la fenêtre ci-dessous s'affiche. Elle permet à l'utilisateur de modifier le paramètre relatif au nombre d'itérations

entre migrations, en lui permettant, ainsi, de contrôler l'emplacement des migrations (c'est à dire, sur quelles itérations la migration aura lieu à partir du nombre d'itérations proposé. (voir figure 46)

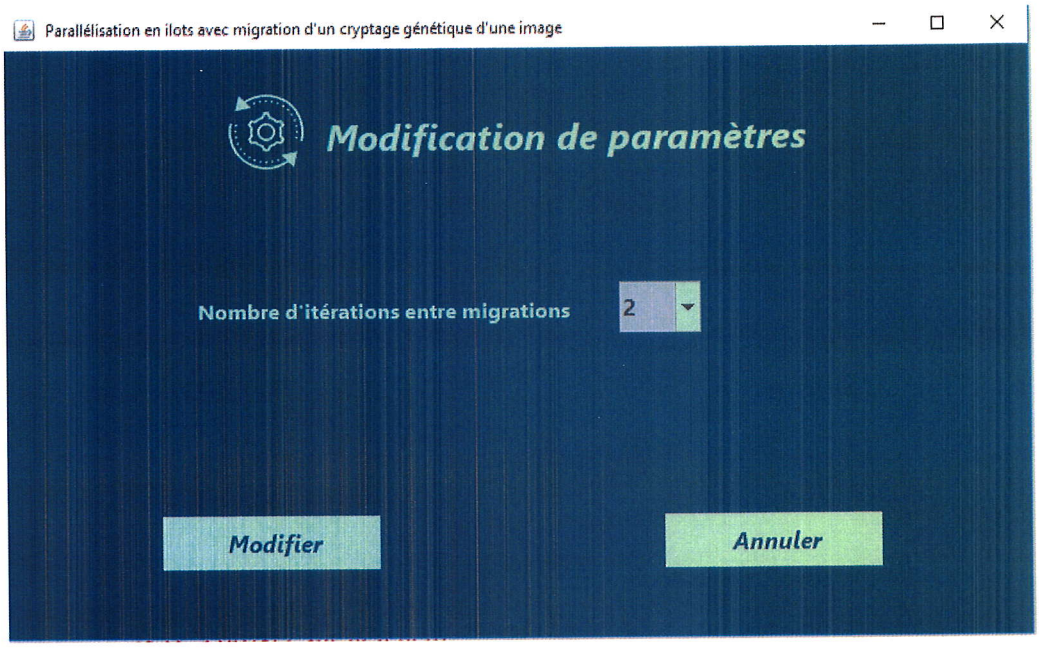


FIGURE 4.46 – La fenêtre « modification de paramètres ».

4.6.2.4 Déchiffrement de l'image chiffrée

La troisième fonctionnalité offerte par la fenêtre « Chiffrement / Déchiffrement d'une image » concerne l'opération de déchiffrement d'une image chiffrée. Une fois validée, la fenêtre illustrée par la figure 47 s'affiche. Cette dernière compte un nombre de bouton chacun réalisant une opération donnée, tel que :

- Le bouton « Télécharger » qui permet de télécharger une image chiffrée.
- Le bouton « Déchiffrer » qui permet de déchiffrer l'image chiffrée précédemment choisie et de visualiser l'image déchiffrée obtenue.
- Le bouton « Retour » qui permet de revenir à la fenêtre précédente (chiffrement / déchiffrement d'une image).
- Un champ permettant à l'utilisateur de proposer un nom pour l'image déchiffrée.

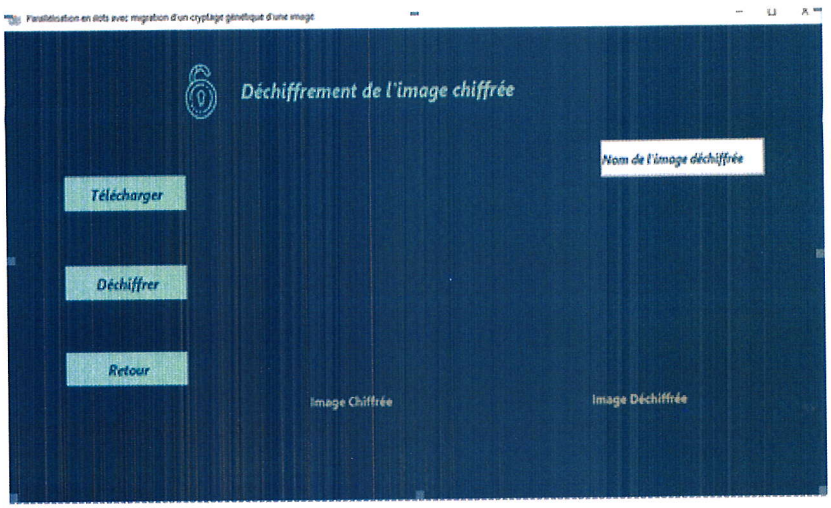


FIGURE 4.47 – la fenêtre « déchiffrement de l'image chiffrée » .

La figure suivante présente un exemple de déchiffrement.

FIGURE 4.48 – Exemple de déchiffrement d'une image chiffrée.

De même que pour le cas de chiffrement, si l'utilisateur tente d'effectuer une opération de déchiffrement avant de choisir (télécharger) une image chiffrée, un message apparaît lui demandant de choisir, tout d'abord, une image à déchiffrer. (Figure 49)

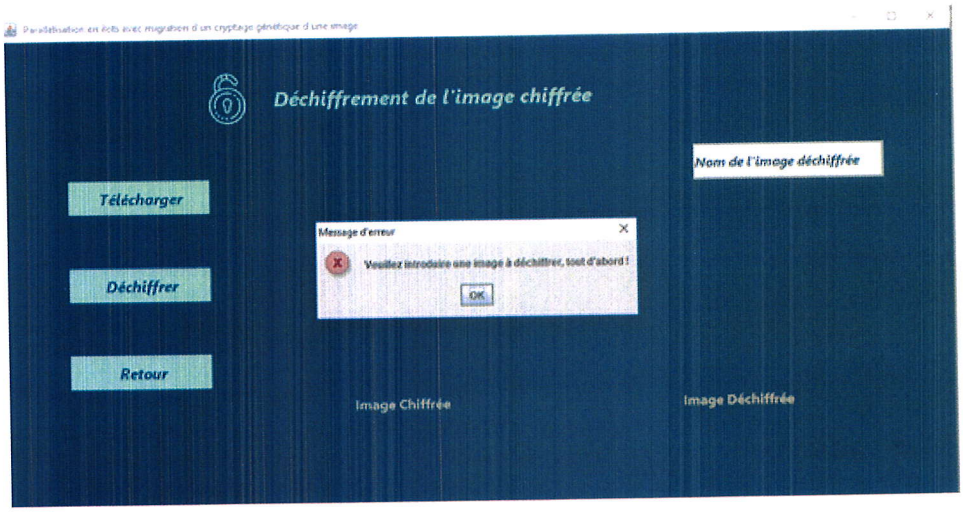


FIGURE 4.49 – le premier message d'erreur.

Le même cas se produira si l'utilisateur ne propose pas de nom pour l'image déchiffrée. (Figure 50)

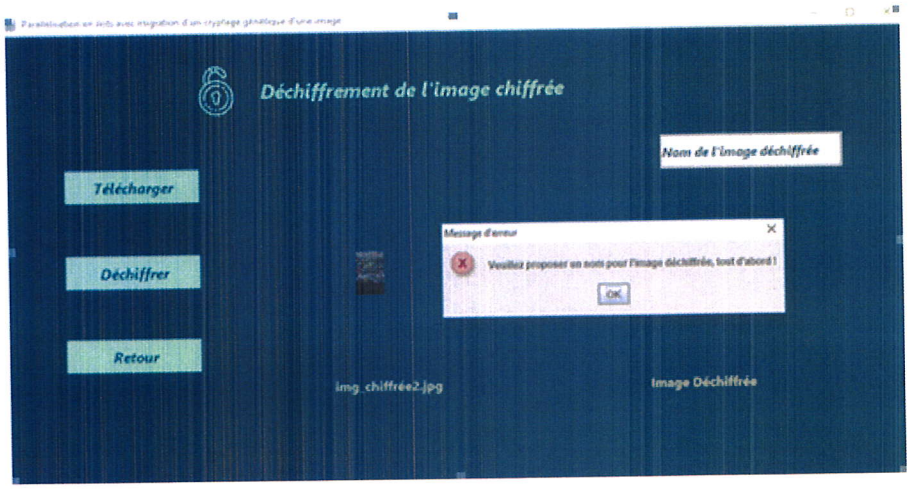


FIGURE 4.50 – Message indiquant la nécessité de proposer un nom pour l'image déchiffrée avant de tenter un déchiffrement.

Si l'image introduite pour déchiffrement a déjà fait l'objet d'une opération de déchiffrement, un avertissement apparaît pour informer l'utilisateur que l'image a déjà été déchiffrée et qu'elle ne peut pas être déchiffrée à nouveau car la clé de déchiffrement de cette version chiffrée a été supprimée lors du premier déchiffrement de l'image. (Figure 51)

FIGURE 4.51 – le troisième message d'erreur.

4.6.2.5 La fenêtre " Informations supplémentaires"

En cliquant sur cette option dans la fenêtre « chiffrement/ déchiffrement d'une image », la fenêtre ci-dessous apparaît. Elle informe l'utilisateur du meilleur paramétrage lui garantissant une meilleure qualité de résultats.

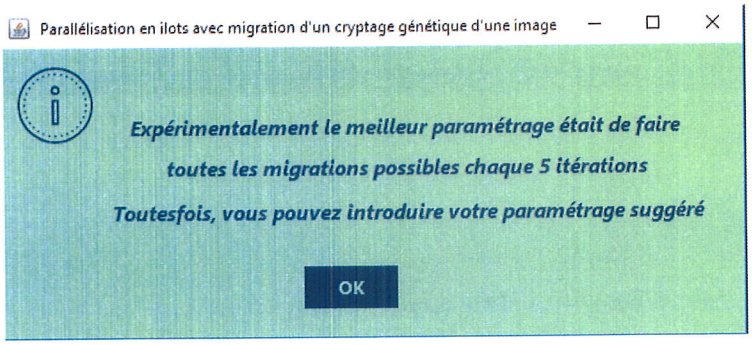


FIGURE 4.52 – la fenêtre de « information supplémentaire ».

En cliquant sur le bouton « OK », l'utilisateur retournera à la fenêtre « chiffrement / déchiffrement d'une image ».

4.7 Conclusion

Dans ce chapitre, nous avons présenté la structure de l'application que nous avons développée à l'aide du langage de programmation JAVA. De même, le paramétrage à adopter a été illustré sur différentes expérimentations pour arriver finalement à conclure sur la suggestion de réaliser toutes les migrations possibles chaque 5 itérations. Toutefois et du coté génétique, nous n'avons que rappeler le meilleur paramétrage génétique réglé à travers le travail [32] car le même type de segmentation a été exploité dans les deux travaux.
Une fois que le paramétrage génétique et celui de migration ont été fixés, nous avons testé la variation de la qualité de solutions envers une augmentation en taille de populations, où un nombre d'individus égale à 10 était le meilleur à recommander.
D'après les résultats obtenus et discutés nous arrivons à la conclusion que PFSPos est l'algorithme le plus rapide que les deux autres versions parallèles PVSPos et PFSPosMig mais PFSPosMig a un pouvoir confusionnel plus important que celui des deux autres algorithmes grâce à la réalisation de toutes les migrations possibles dans une itérations.

Conclusion générale

La cryptographie existe depuis plus de 3 000 ans et pourtant, elle a toujours sa propre place dans le domaine de la sécurité, car les êtres humains ont toujours essayé de cacher leurs informations en temps de paix et de guerre. Les algorithmes génétiques occupent également une place importante en informatique et dans le domaine de la programmation car on sait qu'ils sont l'une des meilleures solutions aux problèmes d'optimisation.

Dans notre travail, nous avons utilisé les algorithmes génétiques afin de résoudre le problème du chiffrement d'images. Pour tirer parti des avantages des algorithmes génétiques parallèles, nous avons choisi d'utiliser le modèle d'îlots avec migrations.

PosESecL2 est une méthode de cryptage génétique proposée par [31] afin de résoudre le problème du chiffrement d'image avec l'utilisation d'un algorithme génétique séquentiel. Dans [32], deux versions parallèles de PosESecL2, PFSPos et PVSPos, ont été proposées afin de réduire le temps de chiffrement d'une image. Les deux versions ont été implémentées à l'aide du modèle d'îlots sans migrations.

Dans notre travail, nous avons implémenté l'algorithme PFSPos en utilisant un modèle d'îlots avec migration, ce qui signifie que nous avons utilisé l'algorithme PFSPos auquel nous avons ajouté l'étape de migration afin de tester son impact sur les performances de l'algorithme ainsi développé (temps de calcul et qualité de l'image chiffrée).

Nous avons commencé par formaliser le problème de cryptage en un problème d'optimisation résoluble par algorithmes génétiques. Une fonction d'évaluation mesurant la performance du processus mis en oeuvre a été, ensuite, définie. Nous avons choisi d'utiliser le même mode de segmentation que celui utilisé par l'algorithme PFPos (segmentation fixe) mais en rajoutant des étapes de migration. Suivant les qualités de l'ensemble d'individus formant les sous-populations évoluant en parallèle, trois types de migration ont été définis : migration du pire au meilleur, migration du meilleur au pire et migration aléatoire.

Dans le dernier chapitre, nous avons présenté la structure de notre application développée à l'aide du langage de programmation JAVA. Ensuite le paramétrage génétique et celui de segmentation ont été rappelés vu que ceux sont les mêmes que ceux adoptés par l'algorithme PFSPos. De même, le réglage du paramètre de migration a fait l'objet d'une étude

expérimentale illustrée sur le même chapitre où deux modes de migrations ont été proposé : réalisation d'une seule migration /itération ou effectuer toutes les migrations possibles / itération, où d'après les résultats expérimentaux obtenus l'adoption du deuxième mode de migration a été recommandée. Il s'agit de réaliser toutes les migrations possibles chaque 5 itérations.

Compte tenu du nombre d'individus, nous avons constaté que l'augmentation du nombre d'individus n'améliorait pas la qualité de l'image. Nous avons donc recommandé de maintenir le nombre d'individus égal à 10 (de même que pour PFSPos). En outre, les résultats expérimentaux obtenus par les trois alternatives parallèles, PFSPos, PVSPos et PFSPosMig, ont montré que PFSPos était le meilleur compte tenu du temps de calcul et de la qualité des images chiffrées obtenues lorsqu'il était appliqué à des images de petites tailles ; cependant, dans le cas d'images de moyennes et de grandes tailles, il est recommandé d'utiliser le PFSPosMig bien qu'il avait le temps de calcul le plus long parmi les autres, mais la qualité des images chiffrées ne peut être ignorée.

Bibliographie

[1] R .Churchhouse. *Codes and Ciphers : Julius Caesar, the Enigma, and the Internet ;Cambridge University Press.* 2001.

[2] K.David. *La guerre des codes secrets.* InterEditions 1980.

[3] Merriam Webster Dictionnary. "https://www.merriam-webster.com/dictionary/cryptography".

[4] La Rousse Dictionnary. "https://www.larousse.fr/encyclopedie/divers/cryptographie/38869".

[5] R. Dumont. Cryptographie et sécurité informatique ; université de liége, 2009-2010.

[6] A. Kerckhoff. La cryptographie militaires. *le Journal des Sciences Militaires*, 1883.

[7] C.Bidan. Cryptographie et cryptanalyse, "https://www.nymphomath.ch/crypto/bibliotheque/PDF/transpcrypto.pdf".

[8] N.Smart. Cryptography : an intoduction ; university of bristol, "https://www.cs.umd.edu/~waa/414-F11/IntroToCrypto.pdf".

[9] K.David. *The Codebreakers : The Story of Secret Writing.* Scribner, 1996.

[10] A.P.Fouque. Introduction a la cryptographie ; universite rennes et institut universitaire de france, "https://www.irisa.fr/prive/sgambs/Intro.pdf".

[11] D. Coppersmith. The data encryption standard (des) and its strength against attacks. *IBM Journal of Research and Development*, 1994.

[12] J.Daemen and V.Rijmen. *The Design of Rijndael : AES-The Advanced Encryption Standard.* Springer, 2002.

[13] G.Paul and S.Maitre. *RC4 Stream cipher and its variants (Discrete Mathematics and its applications).* CRC Press, 2011.

[14] W.Diffie and M.Hellman. New directions in cryptography. *IEEE Transactions on information theory*, 1976.

[15] A.Shamir R.Rivest and L.Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[16] S. Garfinkel. *PGP : Pretty Good Privacy.* O'Reilly Media, 1994.

[17] Cryptographie moderne. "https://www.apprendre-en-ligne.net/crypto/moderne/pgp.pdf".

[18] D.E.Goldberg. *Genetic algorithms in search; optimization and machine learning.* Addison-Wesley Publishing Company, (1989).

[19] C. Bertelle. Chap.2 : Algorithmes génétiques;notes de cours; laboratoire d'informatique du havre, "http://litis.univ-lehavre.fr/~bertelle/iaweb/ia-algo-evol-comp-2p.pdf".

[20] H.J. Bremermann. Hierarchical descriptions of universal spaces and adaptive systems , technical report ora projects 01252 and 08226.ann arbor. *University of Washington,* (1958).

[21] J. H. Holland. The evolution of intelligence. the nervous system as a model of its, environmenttechnical report, no.1, contract no.477(17). *University of Michigan,* (1968).

[22] J. H. Holland. Adaptation in natural and artificial systems, ann arbor. *University of Michigan press,* (1975).

[23] R. Zaghdoud. Hybridation d'algorithme génétique pour les problémes des véhicules intelligents autonomes : applications aux infrastructures portuaires de moyenne taille; these de doctorat; ecole centrale de lille; 2015, "https://tel.archives-ouvertes.fr/tel-01319833/document".

[24] M.A. HAMIDA. Chapitre iii : Introduction aux algorithmes génétiques; notes de cours; université de ouargla, "https://elearn.univ-ouargla.dz/2013-2014/courses/INTRODUCTIONAUXMETHO/document/CH_3.pdf?cidReq=INTRODUCTIONAUXMETHO".

[25] G.Cormier. Chapitre 1 : L'algorithme genetique; notes de cours, "http://www8.umoncton.ca/umcm-cormier_gabriel/SystemesIntelligents/AG.pdf".

[26] A.Kumar. Encoding schemes in genetic algorithm. *International Journal of Advanced Research in IT and Engineering,* page "http://www.garph.co.uk/ijarie/mar2013/1.pdf", (2013).

[27] T.Vallée et M.Yldzoglu. Présentation des algorithmes génétiques et de leurs applications en économie; article; (2001). "http://deptinfo.unice.fr/twiki/pub/Minfo04/IaDecision0405/Prsentationdesalgorithmesgntiquesetdeleursapplicationsenconomie_P.pdf".

[28] W.Benchaib S.Bouhenni S.Si-Moussi F.A.Azoud et M.W.Belahadji. Algorithme génétique sophistiqué et schémas de parallélisation sur une grille de calcul pour la résolution du probléme du sac dos (unbounded knapsack); article. *Ecole Nationale*

Supérieure d'Informatique Algérie; 2016, "http://www.academia.edu/25840289/Algorithme_g%C3%A9n%C3%A9tique_sophistiqu%C3%A9_et_sch%C3%A9mas_de_parall%C3%A9lisation_sur_une_grille_de_calcul_pour_la_r%C3%A9solution_du_probl%C3%A8me_du_sac_%C3%A0_dos_Unbounded_Knapsack_".

[29] M.A.TALEB. Parallélisation d'un algorithme génétique pour le probléme d'ordonnancement sur machine unique avec temps de réglages dépendants de la séquence; mémoire; 2008. *Université du Québec à Chicoutimi*, "https://archipel.uqam.ca/1134/1/M10479.pdf".

[30] khayyam90. Les algorithmes genetiques; 2005. "url"https ://khayyam.developpez.com/articles/algo/g

[31] I.Souici. sécurisation évolutionnaires du transfert d'images; thèsedoctorat; universite badji mokhtar annaba. 2013.

[32] N.laouici et S.Mahmoud. Sécuriser les images à base d'algorithmes génétiques parallèles; mémoire de fin d'étude; université de jijel. 2017.

[33] L.Davis. Applying adaptive algorithms to epistatic domains; proceeding of the international joint conference on artificial intelligence.