N° d'ordre : 08 / 2012 – D / INF

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

Université des Sciences et de la Technologie Houari Boumedienne

Faculté d'Electronique & Informatique

THESE

Présentée pour l'obtention du grade de DOCTORAT

En : INFORMATIQUE

Spécialité : INFORMATIQUE

Par

Mr Mourad DAOUDI

Thème

Approches de résolution par les métaheuristiques de problèmes d'optimisation combinatoire NP-Difficiles

Soutenue publiquement le : 08 / 07 / 2012, devant le jury composé de :

Mme AISSANI – MOKHTARI Aicha	**Professeur à l'USTHB**	**Présidente**
Mr AHMED-NACER Mohamed	**Professeur à l'USTHB**	**Directeur de thèse**
Melle BENATCHBA Karima	**Professeur à l'ESI**	**Examinatrice**
Mme BENBOUZID-SI TAYEB Fatima	**Maître de Conférences/A à l'ESI**	**Examinatrice**
Mr BABA ALI Riad	**Maître de Conférences/A à l'USTHB**	**Examinateur**
Mr BOULIF Menouar	**Maître de Conférences/A à l'U.M.B.B**	**Examinateur**

Approches de résolution par les métaheuristiques de problèmes d'optimisation combinatoire NP-Difficiles

REMERCIEMENTS

Mes remerciements les plus vifs et chaleureux vont à mon directeur de thèse, le professeur *Mohamed AHMED-NACER*, directeur du laboratoire des Systèmes Informatiques (LSI) auquel j'appartiens, pour avoir encadré mon travail de thèse et qui n'a eu de cesse à m'encourager pendant ces longues années.

Je tiens à remercier Mme *Aicha AISSANI MOKHTARI*, Professeur à l'USTHB, d'avoir bien voulu accepter de présider le jury de soutenance. Qu'elle en soit ici remerciée.

Je remercie tout particulièrement Mr *Riad Baba Ali*, Maître de Conférences à l'USTHB, d'avoir bien voulu accepter expertiser mon travail et, aujourd'hui, être membre du jury de soutenance en tant qu'examinateur.

Je remercie Melle *Karima BENATCHBA*, Professeur à l'Ecole Supérieure d'Informatique, qui a bien voulu accepter d'être membre du jury de soutenance en tant qu'examinateur.

Je remercie Mme *Fatima BENBOUZID - SI TAYEB*, Maître de Conférences à l'Ecole Supérieure d'Informatique, qui a bien voulu accepter d'être membre du jury de soutenance en tant qu'examinateur.

Je remercie Mr *Menouar BOULIF*, Maître de Conférences à l'Université de BOUMERDES, qui a bien voulu accepter d'être membre du jury de soutenance en tant qu'examinateur.

J'exprime ici toute ma gratitude au Professeur *Malika Ioualalen-Boukala*, responsable du département Informatique, qui m'a toujours encouragé.

Je tiens à remercier Mr *AbdelMadjid Boukra*, Maître de conférences à l'USTHB, avec qui je partage le bureau, pour toute son aide et son soutien. Qu'il en soit ici remercié.

Je remercie également tous mes collègues enseignants du département Informatique avec qui j'ai plaisir à travailler.

Enfin, je ne remercierai jamais assez toute ma famille, ma femme *Farida* et notre fille *Lylia*, pour leur disponibilité, aide et soutien dans la concrétisation de ce travail.

RESUME

Les métaheuristiques constituent une famille d'algorithmes stochastiques destinés à résoudre des problèmes d'optimisation difficile. Utilisées dans de nombreux domaines de secteurs d'activité différents, ces méthodes présentent l'avantage d'être généralement efficaces. C'est le cas dans le domaine de la sécurité informatique que nous abordons dans le premier volet de la thèse qui traite de la détection d'intrusion dans les systèmes d'information et les réseaux informatiques. Il s'agit plus particulièrement du problème d'optimisation combinatoire NP-difficile de la détection d'intrusion par l'analyse des traces d'audit de sécurité. Ce problème a été introduit par Ludovic Me en1994, étendu par Gomez et Hougen en 2006, puis par Ahmim en 2011. Les différents auteurs utilisent dans leurs travaux les Algorithmes Génétiques comme moteur de détection d'intrusion. Nous proposons deux méthodes de détection d'intrusion utilisant les métaheuristiques Biogeography Based Optimization (BBO) and Harmony Search (HS), qui sont deux algorithms évolutionnaires bien adaptés pour les problèmes d'optimisation sous contraintes. Les tests réalisés sur chacune des deux méthodes de détection proposées ont pour but de régler dans un premier temps les paramètres des métaheuristiques utilisées, puis d'évaluer leurs performances respectives. Les tests sont réalisés d'abord sur un benchmark, puis sur des données générées aléatoirement. Des résultats expérimentaux de détection d'attaques simulées sont donnés. Les deux modèles sont évalués par leur capacité à faire des prévisions correctes, et se sont avérés efficaces et capables de produire des méthodes fiables pour la détection d'intrusion. Une comparaison des deux approches a été réalisée, montrant un temps d'exécution plus court avec la métaheuristique Harmony Search.

L'utilisation de toute métaheuristique nécessite le réglage de ses paramètres. C'est une opération souvent coûteuse en temps et nécessitant un savoir-faire et une expérience de la part de l'utilisateur, d'autant que ces paramètres peuvent avoir une grande influence sur l'efficience et l'efficacité de la recherche de la solution. Utiliser une métaheuristique sans paramètre pourrait palier à ce type de problèmes. C'est dans ce contexte, que nous proposons dans le deuxième volet de notre thèse, une approche de résolution d'un problème d'optimisation combinatoire NP-difficile, à savoir le Problème du Voyageur de Commerce, par une métaheuristique sans paramètre. Cette dernière est adaptée de la métaheuristique sans paramètre « TRIBES » qui a été développée par Maurice Clerc [Clerc, 2003] pour résoudre des problèmes continus. Ainsi, une extension de TRIBES à la résolution de problèmes discrets est développée et adaptée à la résolution du Problème du Voyageur de Commerce. Cela est rendu possible par l'introduction d'une distance dans l'espace des solutions défini par l'ensemble des cycles Hamiltoniens dans le graphe représentatif du PVC. Les différents mécanismes développés dans TRIBES, comme le processus de génération des particules ou les stratégies de déplacement développées dans les phases d'adaptation structurelle et comportementale de TRIBES sont modifiés. Nous avons observé le bon comportement de TRIBES discrétisée sur plusieurs tests. Une comparaison avec un algorithme génétique a montré des résultats compétitifs.

Mots-clés: Optimisation combinatoire, métaheuristiques, sécurité informatique, audit de sécurité, Biogeography Based Optimization, Harmony Search, Tribes.

ABSTRACT

Metaheuristics are a family of stochastic algorithms to solve hard optimization problems. Used in many fields of different aria, these methods have the advantage of being generally effective. This is the case in the field of computer security that we address in the first part of the thesis and concerns intrusion detection in information systems and computer networks. More specifically, we treat of a particular NP-Hard combinatorial optimization problem that is the security audit trail analysis, initiated by Ludovic Me in 1994, and extended by Gomez and Hougen in 2006, and Ahmim in 2011. They all use genetic algorithms as intrusion detection engine in their works. We suggest two intrusion detection methods using the metaheuristics Biogeography Based Optimization (BBO) and Harmony Search (HS). They are evolutionary algorithms, well suited for constrained optimization problems. Tests are performed to fix the metaheuristics parameters values and to evaluate the proposed approaches. Results on simulated intrusions' detection are given. The effectiveness of the models is evaluated by their ability to make correct predictions. They are effective and capable of producing reliable methods for intrusion detection. An important result was the consistency of detection results, irrespective to the number of attacks actually present in the analysed audit file. A comparison of both approaches is made. The execution time in the HS-based method is far better.

However the calculation of execution time does not take into consideration the time required to determine the values of the parameters used. Indeed, as with any metaheuristic, the parameter setting is time consuming and requires expertise and experience from the user, especially since these parameters can have a great influence on the efficiency and effectiveness of the search for the solution. Use a metaheuristic with no parameters could compensate for such problems. It is in this context, we propose in the second part of our thesis, an approach to solving the well-known NP-hard combinatorial optimization problem, namely the Travelling Salesman Problem, using a parameter-free metaheuristic. This latter is adapted from "TRIBES", a parameter-free metaheuristic, developed by Maurice Clerc in 2003 to solve continuous problems. Our work is concerned with an extension of TRIBES algorithm to solve discrete problems. An adaptation to solve TSP problem is made possible when defining a distance in the search space of the Hamiltonian cycles in the TSP representative graph. The different mechanisms developed in TRIBES are modified, such as the particles generating process or the displacement strategies developed during the structural and behavioral adaptation phases in TRIBES. Experimentation results are given and a comparison with a Genetic Algorithm showed competitive results.

Key-words: Combinatory optimization, metaheuristics, computer security, security audit , Biogeography Based Optimization, Harmony Search, Tribes.

Table des matières

Liste des figures

Liste des algorithmes

INTRODUCTION GENERALE

L'optimisation combinatoire est une discipline qui présente un intérêt majeur tant sur le plan théorique que pratique. Sur le plan théorique, de nombreux concepts très novateurs ont été proposés lors du siècle dernier. Aujourd'hui, ces concepts continuent d'être enrichis par une bibliographie très riche et très dense. Sur le plan pratique, et dans de nombreux domaines différents, cette discipline connaît un regain d'intérêt majeur. De nombreux problèmes d'intérêt rencontrés dans la pratique sont des problèmes d'optimisation combinatoire NP-difficiles, et de ce fait intraitables par des méthodes classiques. L'énumération de toutes les solutions réalisables pour trouver la meilleure solution s'avère impossible même pour un problème de taille moyenne. L'impossibilité de pouvoir résoudre la grande diversité de problèmes d'optimisation de grandes dimensions par des méthodes classiques, dites exactes, justifie l'emploi de méthodes moins exigeantes et pouvant être plus efficaces, telles que les métaheuristiques. La plupart d'entre elles utilisent des processus aléatoires comme moyens de récolter les informations et de faire face à des problèmes comme l'explosion combinatoire. Même si elles ne garantissent pas l'optimalité des résultats, elles permettent l'obtention de « bonnes » solutions dans un temps raisonnable. Nous considérons dans cette thèse des approches de résolution par des métaheuristiques de deux problèmes d'optimisation combinatoire NP-difficiles.

.

La détection d'intrusion par l'analyse des traces d'audit de sécurité dans les systèmes d'information et les réseaux informatiques constitue notre premier centre d'intérêt. En effet, leur surveillance régulière est primordiale pour une détection efficace des attaques de plus en plus nombreuses et sophistiquées de la part de pirates toujours à la recherche de nouvelles failles à exploiter et de nouveaux moyens pour infiltrer, endommager ou dérober les données critiques ou simplement interrompre l'activité des entreprises. Aussi découvrir rapidement une intrusion ou une tentative d'intrusion permet d'éviter ou de limiter l'étendue des dommages. Les systèmes de détection d'intrusion constituent l'une des premières approches au problème de la sécurité informatique. L'étude de méthodes efficaces pour détecter les intrusions dans les fichiers de traces d'audit de sécurité est un élément important du vaste effort visant à améliorer les systèmes de détection d'intrusion. La majorité de ces méthodes sont basées sur des techniques d'intelligence artificielle, incluant les réseaux de neurones, les systèmes immunitaires et les algorithmes génétiques.

Nous considérons ainsi le problème de la détection d'intrusion par l'analyse des traces d'audit de sécurité. C'est une approche dite réactive, qui consiste à essayer de détecter les attaques au plutôt afin de réagir rapidement et éviter ainsi que de sérieux dommages soient causés. Ce mécanisme de détection (à postériori) génère une quantité importante d'information, qui peut vite s'avérer indigeste pour les administrateurs réseau et l'information utile qu'on peut tirer en est considérablement limitée.

Afin de pallier à cette difficulté, nous utilisons une méthode d'analyse simplifiée des fichiers d'audit. Cette approche consiste alors à rechercher des scénarios d'attaques prédéfinis dans les traces d'audit. Chaque attaque est définie par le seuil du nombre d'occurrence de chaque évènement auditable. Pour qu'une attaque ait lieu, ces seuils doivent se trouver dans les traces d'audit. C'est une approche qui se formalise en un problème d'optimisation combinatoire NP-difficile. Nous proposons alors deux méthodes de résolution utilisant les métaheuristiques : « Biogeography Based Optimization » (BBO) et « Harmony Search » (HS), de la famille des algorithmes évolutionnaires, avérées efficaces pour les problèmes d'optimisation sous contraintes.

Dans le deuxième volet de notre thèse, nous considérons le problème classique d'optimisation combinatoire: le Problème du Voyageur de Commerce (PVC). Il consiste à minimiser la longueur de la tournée d'un « voyageur de commerce », qui doit visiter n villes avant de retourner à sa ville de départ. C'est un problème NP-difficile, objet d'un très grand nombre de travaux dans la littérature, utilisant différentes métaheuristiques pour sa résolution. Cependant, toutes sont confrontées au problème du réglage des paramètres. En effet, comme pour la plupart des métaheuristiques, des valeurs de nombreux paramètres doivent être déterminées. Ces valeurs peuvent avoir une grande influence sur l'efficience et l'efficacité de la recherche de la solution du problème à résoudre. Les valeurs optimales pour les paramètres dépendent essentiellement du problème et même de l'instance à traiter ainsi que du temps de recherche que l'utilisateur veut passer à résoudre le problème. Par ailleurs, les paramètres sont souvent corrélés lorsqu'il s'agit de problèmes réels, ce qui rend le choix des paramètres plus difficile. Ainsi, l'exploration d'une combinaison « optimale » de ces valeurs peut être difficile et consommer beaucoup de temps. Pour s'affranchir de ce type de réglage, des algorithmes dits « adaptatifs » ont été développés. Avec ces algorithmes, les valeurs des paramètres ne sont plus figées, mais sont modifiées, en fonction des résultats collectés durant le processus de recherche.

C'est dans ce contexte, que nous proposons une approche de résolution par une métaheuristique sans paramètre du problème du PVC. La métaheuristique sans paramètre utilisée est adaptée de la métaheuristique « TRIBES » qui a été développée par Maurice Clerc [Clerc, 2003] pour résoudre des problèmes continus. Il y'a lieu alors d'étendre « TRIBES » à la résolution de problèmes discrets pour résoudre le PVC. Cela est rendu possible par l'introduction d'une distance dans l'espace des solutions défini par l'ensemble des cycles Hamiltoniens dans le graphe représentatif du PVC. Les différents mécanismes développés dans TRIBES, comme le processus de génération des particules ou les stratégies de déplacement développées dans les phases d'adaptation structurelle et comportementale de TRIBES sont modifiés.

En plus de cette introduction, le document comprend 4 chapitres. Dans le chapitre 1, nous explicitons des notions d'optimisation combinatoire ainsi que des différentes méthodes de résolution de problèmes d'OC. Le chapitre suivant présente un état de l'art des métaheuristiques d'optimisation.

Une étude de quelques métaheuristiques souvent utilisées et de leurs caractéristiques est présentée. Un intérêt particulier est porté aux méthodes d'optimisation utilisées dans les chapitres 3 et 4, telles que BBO, HS et TRIBES.

Les chapitres 3 et 4 portent sur notre contribution. Le chapitre 3 traite de l'apport des métaheuristiques (BBO et HS) à la résolution du problème d'optimisation combinatoire particulier au domaine de la sécurité informatique, à savoir le problème d'analyse des traces d'audit de sécurité. Le chapitre 4 traite de la résolution du PVC en utilisant une métaheuristique sans paramètre développée à partir de la métaheuristique sans paramètre « TRIBES » conçue pour la résolution de problèmes continus. Les résultats de simulation ainsi que leurs interprétations sont présentés en fin des chapitres 3 et 4. Une conclusion générale clôture le document.

Chapitre 1

Optimisation combinatoire

1.1 Introduction

L'optimisation combinatoire (OC) est une discipline combinant différentes techniques des mathématiques discrètes et de l'informatique pour résoudre des problèmes d'optimisation. Elle traite de problèmes issus de nombreux secteurs d'activité, multiples et variés, aussi bien théoriques que pratiques, qui consistent à trouver le meilleur entre un nombre fini de choix. Il s'exprime par une fonction (dite de coût) avec ou sans contraintes à minimiser (ou à maximiser) sur un ensemble fini [Papadimitriou et Steiglitz, 1982].

Les problèmes d'OC sont souvent faciles à définir mais difficiles à résoudre. Le degré de difficulté à résoudre un problème avec un certain algorithme spécifique est étroitement lié à sa complexité de calcul, c'est à dire, la quantité de ressources nécessaire à sa résolution, telles que le temps et la mémoire. Les problèmes peuvent alors être divisés en classes de complexité. De nombreux problèmes d'optimisation combinatoire sont NP-Complets, aucun algorithme en mesure de les résoudre en temps polynomial sur un ordinateur déterministe n'existe à ce jour [Garey et Johnson, 1979]. Une approche pour l'obtention de solutions quasi-optimales dans un délai raisonnable est l'utilisation des métaheuristiques qui sont des procédures d'optimisation stochastique.

La suite de ce chapitre consiste en la présentation de notions d'optimisation combinatoire, ainsi que de méthodes de résolution de problèmes d'optimisation combinatoire.

1.2 Problème d'Optimisation combinatoire

1.2.1 Définitions

En mathématiques, l'optimisation recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction, avec ou sans contraintes. En théorie, un problème d'optimisation combinatoire se définit par l'ensemble de ses instances, souvent infiniment nombreuses. Dans la pratique, le problème se ramène à résoudre numériquement l'une de ces instances, par un procédé algorithmique. A chaque instance du problème est associé un ensemble discret de solutions *S*, un sous-ensemble *X* de *S* représentant les solutions admissibles (réalisables), ainsi qu'une fonction f appelée fonction objectif ou encore fonction coût (fitness). La fonction f assigne à chaque solution s Є X le

nombre f(s). Résoudre un problème d'optimisation combinatoire consiste alors à trouver une solution s Є X qui minimise la valeur de la fonction objectif f [Papadimitriou et Steiglitz, 1982].

Un problème d'optimisation combinatoire peut être également défini formellement à partir d'un triplet (E, p, f) tel que :

– E est un ensemble discret appelé espace des solutions ou encore espace de recherche ;

– p est un prédicat sur E, i.e. une fonction de E dans {Vrai, Faux} ;

– f : E →R associe à tout élément x Є E un coût f (x), f est la fonction objectif ou fonction de coût.

L'ensemble des solutions admissibles du problème, noté Ea, est défini tel que

Ea = {x Є E tel que p(x) est Vrai}

Il s'agit de trouver l'élément x* Є Ea qui minimise f :

$$f(x^*) = \min_{x \in Ea} f(x)$$

Remarque : Le problème d'optimisation combinatoire qui consiste à chercher un élément maximum est de même nature :

$$\text{Max } f(x) = -\min_{x \in Ea} [-f(x)]$$

Notons une notion importante qu'il s'agit de préciser dans la pratique quand on résout un problème d'optimisation combinatoire à savoir la notion de voisinage. À chaque solution ***s*** du problème, on associe un sous-ensemble V(**s**) de solutions. Ce sous-ensemble peut être statique ou dynamique (dépendre du temps *t* ou plus précisément de l'itération à laquelle on se trouve). En optimisation combinatoire, il arrive souvent que les solutions peuvent s'écrire sous la forme d'une séquence finie d'éléments. Etant donnée une solution s, un voisinage V(s) de s peut alors être défini comme l'ensemble des modifications élémentaires que l'on peut appliquer à s, par exemple l'ensemble des permutations. A titre d'exemple, considérons l'exemple du problème du voyageur de commerce, qui consiste à minimiser la longueur de la tournée d'un « voyageur de commerce », qui doit visiter n villes, avant de retourner à sa ville de départ. Partant d'une tournée A – B – C – D – E, un voisinage pourra se définir comme l'ensemble des inversions de 2 villes sur le parcours (une inversion possible : A – B – D – C – E), ou bien comme l'ensemble des déplacements unitaires (un déplacement possible : A – B – E – C – D).

1.2.2 Optimum global, optimum local

Soit un problème d'optimisation combinatoire (E, p, f) et Ea l'ensemble des solutions admissibles du problème induit par p. Soit x* Є Ea.

- Si l'on peut prouver que $\forall x \in Ea$, $f(x^*) \leq f(x)$, alors on dira que x^* est l'optimum (minimum) global du problème ;
- S'il existe un ensemble V C Ea, contenant x^*, et au moins deux éléments, tel que : $\forall x \in V$, $f(x^*) \leq f(x)$, alors on dira que x^*est un optimum (minimum) local du problème.

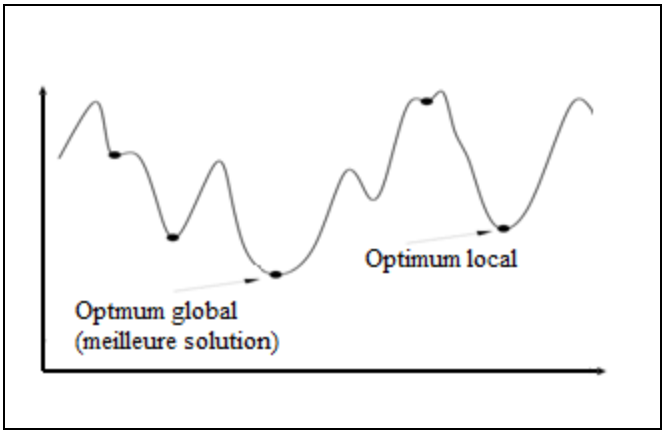


Figure 1.1 - Optimum global, optimum local

1.2.3 Convergence prématurée

Les algorithmes d'optimisation sont guidés par des fonctions objectifs. Des problèmes peuvent surgir lors du processus d'optimisation, et cela en raison de caractéristiques particulières de la fonction objectif. On dit que le processus d'optimisation a convergé prématurément vers un optimum local s'il n'est plus en mesure d'explorer des parties de l'espace de recherche autres que la région en cours d'examen, alors qu'il existe une autre région qui contient une meilleure solution [Schaffer et al., 1990], [Ursem, 2003]. Il est alors souvent impossible de déterminer si la meilleure solution actuellement connue est située sur un optimum local ou global et donc, si la convergence est acceptable. En d'autres termes, il n'est généralement pas clair si le processus d'optimisation peut être arrêté, ou alors s'il doit se concentrer sur l'amélioration de l'optimum courant, ou s'il devrait examiner d'autres parties de l'espace de recherche. Certaines méthodes d'optimisation, qui partent d'une solution initiale et qui l'améliorent en explorant son voisinage immédiat, présentent l'inconvénient de s'arrêter au premier optimum local trouvé. Cela peut, bien sûr, devenir problématique s'il existe plusieurs optima (locaux), c'est à dire, le problème est multimodal (une fonction mathématique est multimodale si elle a de multiples maxima ou minima [Shekel, 1971]). La convergence prématurée aux minima locaux est présentée comme l'un des principaux symptômes d'échec des processus d'optimisation. L'existence de plusieurs optima globaux n'est pas en soi problématique et la découverte de seulement un sous-ensemble d'entre eux peut encore être considérée comme un succès dans de nombreux cas. La présence de nombreux optima locaux, cependant, pose problème. La Figure 1.2 illustre un exemple de

la convergence prématurée. Comme nous le verrons plus loin, les méthodes de résolution contiennent souvent une technique ou une astuce qui permettent d'éviter de se retrouver piégé dans ces minima locaux, en explorant davantage tout l'espace de recherche, de façon à augmenter la probabilité de rencontrer le minimum global.

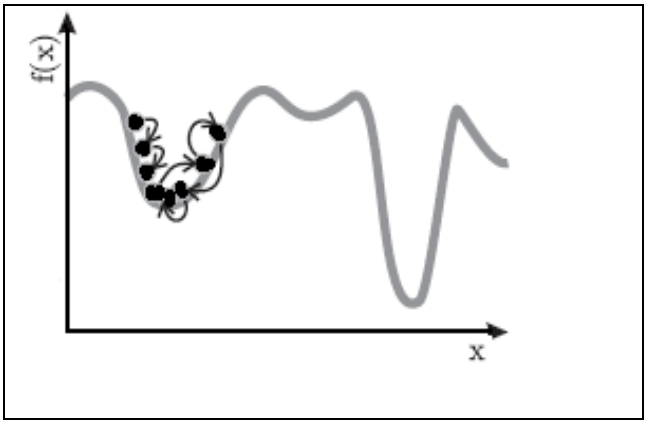


Figure 1.2 - Convergence prématurée (minimisation)

Avant de nous intéresser aux méthodes de résolution, on notera que le degré de difficulté à résoudre un problème avec un algorithme spécifique est étroitement lié à sa complexité de calcul, c'est à dire, le montant des ressources nécessaires pour ce faire, telles que le temps et la mémoire. La complexité de calcul dépend du nombre d'éléments d'entrée nécessaires pour appliquer l'algorithme. Cette dépendance est souvent exprimée sous la forme de limites approximatives avec les notations introduites par Bachmann [Bachmann, 1894] et rendues populaires par Landau [Landau, 1909]. La théorie de la complexité traite principalement de l'efficacité des algorithmes et des classes de problèmes pour lesquels des algorithmes efficaces sont possibles. Il est important de savoir qu'un problème puisse être résolu par un algorithme en temps polynomial ou non. Une discussion complète sur le sujet peut être trouvée dans le livre de Garey et Johnson [Garey et Johnson, 1979].

1.3 Méthodes de résolution

1.3.1 Introduction

De par l'importance des problèmes d'optimisation combinatoire, de nombreux chercheurs ont porté un grand intérêt à leur résolution. Plusieurs méthodes ont ainsi été développées. Pendant longtemps, la recherche s'est orientée vers la proposition d'algorithmes exacts pour des cas particuliers polynomiaux. Ensuite, l'apparition des heuristiques a permis de trouver des solutions en général de bonne qualité pour résoudre les problèmes. En même temps, les méthodes de type "séparation et évaluation" ont aidé à résoudre des problèmes de manière optimale, mais souvent pour des instances de petite taille. Parmi les méthodes approchées, on retrouve les heuristiques et les métaheuristiques.

Ces dernières peuvent être réparties dans deux classes : les métaheuristiques à solution courante unique et les métaheuristiques à population (ensemble de solutions). Nous suivrons cette classification dans la suite de la présentation, comme schématisée dans la figure 1.3.

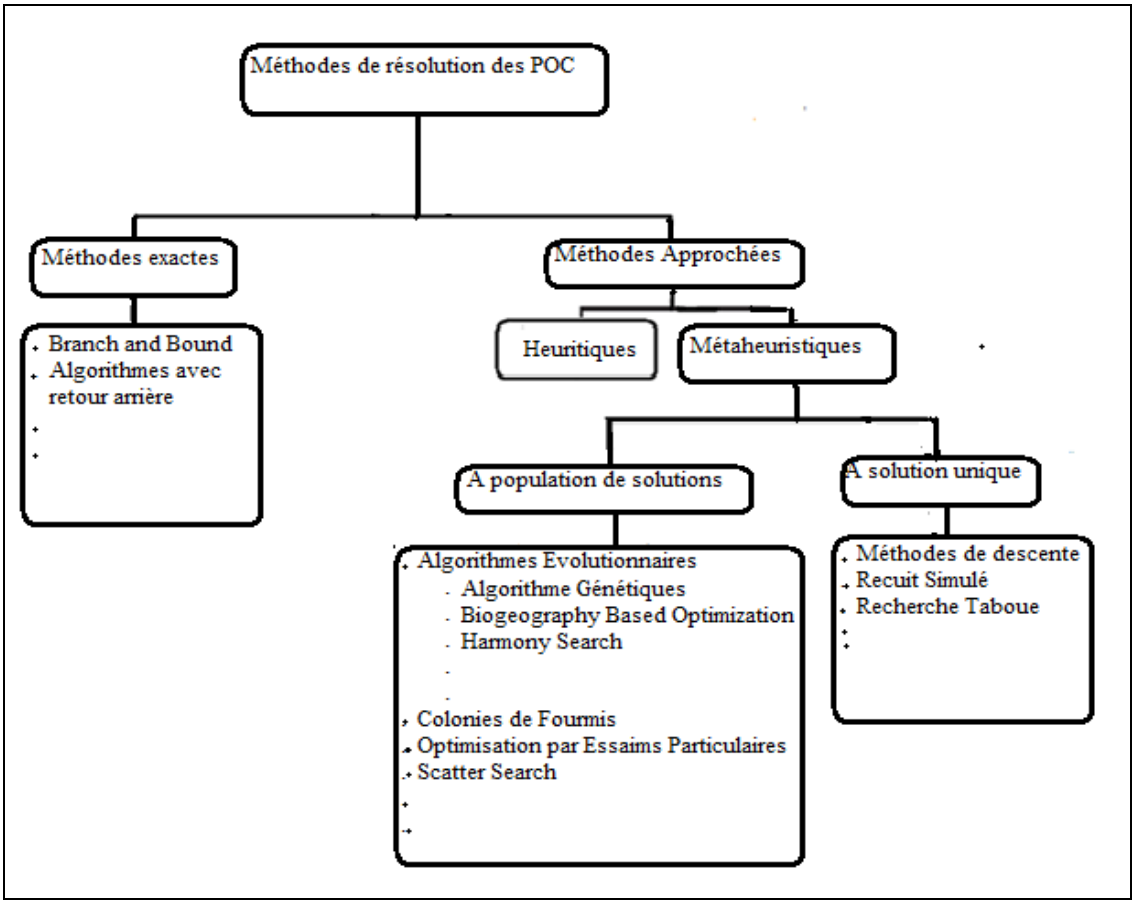


Figure 1.3 - Taxonomie des méthodes de résolution des problèmes d'OC

1.3.2 Méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de recherche. Le recours à des méthodes exactes permet de bénéficier de résultats théoriques forts accompagnant les notions de convergence et d'optimalité globale. Elles peuvent être très efficaces pour des problèmes de petite taille. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calcul de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes (dites aussi déterministes, i.e. : les méthodes qui explorent de manière déterministes l'espace de recherche), figurent : la méthode de séparation et évaluation (Branch and Bound), la programmation dynamique, les méthodes de relaxation Lagrangiennes, les méthodes de programmation linéaire ainsi que les méthodes de programmation en nombres entiers, telles que

branch-and-cut [Padberg et Rinaldi], branch-and-price, and branch-and-cut-and-price [Puchinger et Raid, 2005], [Nemhauser et Wolsey, 1988]

Nous donnons ci-dessous brièvement le principe de fonctionnement de la méthode Branch&Bound.

1.3.2.1 Méthodes de Séparation et Evaluation (Branch &Bound)

Les méthodes de séparation et évaluation permettent la construction d'une arborescence dont chacun des sommets représente un sous problème et les arcs issus d'un même sommet représentent une décomposition possible du problème situé au sommet de l'arbre en sous-problèmes de taille réduite. Cette arborescence est explorée de façon à éviter les branches ne contenant pas de solutions réalisables et les branches n'amenant pas à des solutions meilleures que la solution courante. Pour cela, on se base sur l'utilisation de bornes (inférieure et supérieure) qui représentent l'estimation de la valeur de la meilleure solution obtenue de la branche parcourue.

Ces méthodes de séparation et évaluation reposent sur quatre points essentiels :

- La séparation qui permet de décomposer un problème en le partitionnant successivement en sous problèmes de taille réduite, et ce jusqu'à l'obtention d'ensembles suffisamment petits pour pouvoir énumérer toutes les solutions qu'ils contiennent. Le nombre de séparations doit être fini.
- L'évaluation qui associe une borne au critère d'optimisation sur l'ensemble des solutions d'un sous-problème (borne supérieure dans le cas d'une minimisation). Cette borne permet d'éviter l'exploration d'un sommet dont la solution partielle correspondante possède une valeur du critère dépassant la borne.
- Le sondage qui permet de déterminer si un sommet est terminal (i-e : il ne contient pas de solutions admissibles, ou c'est une solution optimale, ou on peut obtenir polynomialement la solution optimale de ce sous-problème), ou il mérite d'être séparé.
- La sélection qui décrit comment choisir les sous-problèmes à séparer, lorsque plusieurs sont candidats. On peut distinguer deux stratégies d'exploration, celle qui favorise les meilleurs d'abord appelée aussi procédure par séparation et évaluation progressive, et celle de type profondeur d'abord et retour arrière appelée aussi procédure par séparation et évaluation séquentielle.

1.3.3 Méthodes approchées

Contrairement aux méthodes dites exactes, les méthodes approchées ne procurent pas forcément une solution optimale, mais seulement une bonne solution en fonction du temps disponible. Leur capacité à solutionner un problème difficile à résoudre par des méthodes exactes est nuancée par le fait

qu'elles n'offrent aucune garantie quant à la qualité de la solution trouvée. Ce défaut se révèle mineur lorsque seule une approximation de la solution optimale est recherchée. Elles améliorent l'exploration de l'espace des solutions d'un problème donné, en guidant le processus dans sa recherche de solutions optimales. Parmi ces méthodes, on trouve les heuristiques et les métaheuristiques [Dréo et al., 2006].

Les heuristiques (le mot heuristique vient du verbe grec heuriskein, qui signifie « trouver ») peuvent être conçues comme des algorithmes simples souvent basés sur l'intuition, conçues pour résoudre des problèmes particuliers.

Un exemple de méthode heuristique pour le problème du voyageur de commerce est :

Partir de la ville 1
Répéter
    Aller à la ville la plus proche non encore visitée
Tant qu'il reste des villes non visitées

Lorsqu'une heuristique est adaptable sur des types de problèmes variés, sans changements majeurs de l'algorithme, le terme de "métaheuristiques" est employé (le mot Méta vient du grec μετα qui signifie « qui dépasse, englobe »). Les heuristiques ou les métaheuristiques exploitent généralement des processus aléatoires dans l'exploration de l'espace de recherche pour faire face à l'explosion combinatoire engendrée par l'utilisation de méthodes exactes. En plus de cette base stochastique, les méta-heuristiques sont le plus souvent itératives, ainsi le même processus de recherche est répété lors de la résolution. Leur principal intérêt provient justement de leur capacité à éviter les minima locaux, en admettant par exemple une dégradation de la fonction objectif au cours de leur progression. Elles constituent ainsi une famille de techniques stochastiques, devenue ces dernières décennies un domaine de recherche actif. Le premier travail dans ce domaine a commencé il y a environ un demi-siècle [Robbins et Monro, 1951], [Friedberg,1958], [Bledsoe and Browning, 1959] et [Bremermann, 1962]. Par la suite, un grand nombre de méthodes ont été proposées, utilisant un ensemble limité de concepts dans leur quête de l'optimal, et sont d'inspiration diverse, telle que la physique (recuit simulé), la génétique (algorithmes évolutionnaires), l'éthologie (colonies de fourmis), etc.

Le chapitre 2 suivant reprend plus en détail cette famille de méthodes, présentant les caractéristiques de différentes métaheuristiques avec leurs avantages et inconvénients.

1.4 Conclusion

Il est important de savoir qu'un problème d'optimisation puisse être résolu par un algorithme en temps polynomial ou non avant de décider de la nature exacte ou approchée de la méthode de

résolution à utiliser. De fait, plusieurs problèmes classiques (de taille raisonnable) peuvent être résolus de manière exacte en un laps de temps acceptable. Cependant, de nombreux problèmes de grand intérêt aussi bien théoriques que pratiques sont NP-difficiles. On ne connait pas d'algorithme temps polynomial non déterministe qui les résolvent efficacement sauf si P=NP. Ainsi, pour ce type de problèmes, les méthodes exactes permettent seulement de résoudre des instances de petite ou moyenne taille. L'utilisation de méthodes approchées devient obligatoire. Une présentation des plus brèves en a été faite dans ce chapitre. Le chapitre suivant est consacré aux métaheuristiques d'optimisation, avec un intérêt particulier porté aux méthodes utilisées dans les travaux présentés aux chapitres 3 et 4.

Chapitre 2

Métaheuristiques d'optimisation : état de l'art

2.1 Introduction

Les métaheuristiques constituent une classe de méthodes qui fournissent des solutions de bonne qualité en temps raisonnable à des problèmes combinatoires réputés difficiles pour lesquels on ne connait pas de méthode classique plus efficace. Elles sont formellement définies comme des processus de génération itératifs qui guident une heuristique subordonnée en combinant intelligemment différents concepts pour l'exploration et l'exploitation de l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer les informations afin de trouver efficacement des solutions quasi-optimales. [Osman et Laporte, 1996].

Blum et Roli [Blum et Roli, 2003] caractérisent les métaheuristiques de la façon suivante:

- Les métaheuristiques sont des stratégies qui «guident» le processus de recherche.
- Le but est d'explorer efficacement l'espace de recherche afin de trouver des solutions quasi-optimales.
- Les techniques utilisées vont de simples procédures de recherche locale aux processus complexes d'apprentissage.
- Les algorithmes sont approximatifs et généralement non-déterministes.
- Ils peuvent incorporer des mécanismes pour éviter d'être piégés dans des zones confinées de l'espace de recherche.
- Les concepts de base des métaheuristiques permettent une description du niveau abstrait.
- Les métaheuristiques ne sont pas spécifiques au problème.
- Les métaheuristiques peuvent faire usage du domaine spécifique des connaissances sous la forme d'heuristiques qui sont contrôlées par la stratégie de niveau supérieur.

Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction, en évaluant une certaine fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème afin d'en trouver une approximation de la meilleure solution d'une manière proche des algorithmes d'approximation. En outre, elles n'ont pas de critère d'arrêt naturel. Elles utilisent des critères d'arrêt tels le temps de calcul maximum ou le nombre d'itérations.

Comme les heuristiques, les métaheuristiques n'offrent généralement pas de garantie d'optimalité, bien qu'on ait pu démontrer la convergence de certaines d'entre elles. Non déterministe, elles incorporent souvent un principe stochastique pour surmonter l'explosion combinatoire. L'expérience accumulée durant la recherche de l'optimum est souvent mise à profit pour mieux guider la suite du processus de recherche. Des détériorations de la solution courante peuvent être admises, de même que la génération de nouvelles solutions initiales pour la recherche locale d'une façon plus intelligente que simplement fournir des solutions initiales aléatoires.

2.2 Principe des métaheuristiques

Les métaheuristiques sont basées sur un nombre de concepts limité. La représentation ou codage de la solution ainsi que la définition de la fonction d'évaluation sont deux éléments essentiels dans la conception des métaheuristiques. De même, l'intensification et la diversification dans la recherche de la solution constituent deux concepts fondamentaux.

2.2.1 Représentation des solutions

La représentation de la solution doit être adaptée et pertinente pour le problème d'optimisation considéré, car la qualité d'une représentation a une influence considérable sur l'efficacité des opérateurs de recherche appliqués sur cette représentation. Quatre codages majeurs dans la littérature peuvent être soulignés: le codage binaire (par exemple : problème du sac à dos, problème de satisfiabilité), vecteur de valeurs discrètes (problème de la localisation, problème d'affectation), permutation (par exemple problème du voyageur de commerce, les problèmes d'ordonnancement) et le vecteur de valeurs réelles (par exemple des fonctions continues) [Luong, 2011].

2.2.2 Fonction d'évaluation

La fonction objectif f formule le but à atteindre. Elle associe à chaque solution de l'espace de recherche une valeur réelle qui donne la qualité ou la fitness de la solution. Puis, elle représente une valeur absolue et permet un classement complet de toutes les solutions de l'espace de recherche. La fonction objectif est un élément essentiel dans la conception d'une métaheuristique. Elle guide la recherche vers de «bonnes» solutions de l'espace de recherche. Si la fonction objectif est mal définie, elle peut conduire à des solutions non acceptables quelle que soit la métaheuristique utilisée.

2.2.3 Intensification et diversification

Identifier rapidement les régions dans l'espace de recherche avec des solutions de haute qualité et ne pas perdre trop de temps dans les régions de l'espace de recherche qui sont soit déjà explorées ou qui ne fournissent pas de solutions de haute qualité constituent également une préoccupation majeure

dans la conception des métaheuristiques. Deux concepts fondamentaux sont définis : l'intensification (exploitation de l'expérience de recherche accumulée : on recherche les meilleures solutions dans la région de l'espace de recherche en cours d'analyse) et la diversification (exploration de l'espace de recherche quand il est possible de déterminer que la recherche se concentre sur de mauvaises zones de l'espace de recherche). Un compromis entre ces deux concepts doit être assuré.

La convergence prématurée aux minima locaux est souvent présentée comme l'un des principaux symptômes de l'échec du processus d'optimisation, et l'une des raisons de cet échec est le manque de diversité des solutions dans l'espace de recherche [Weise et al., 2008]. Dans certaines métaheuristiques à base de population de solutions, le maintien d'un ensemble de solutions candidates diversifié est d'une grande importance. Perdre la diversité, c'est approcher un état où toutes les solutions candidates en cours d'investigation sont semblables les unes aux autres, c'est-à-dire en fait qu'il ya convergence. Préserver la diversité est directement liée au maintien d'un bon équilibre entre l'intensification et la diversification [Paenke et al.,2007] . Ceci a fait l'objet de plusieurs travaux [Weise et al., 2008] dans de nombreux domaines, tels que les Algorithmes Génétiques, les Algorithmes Evolutionnaires, la Programmation Génétique, la Recherché Tabou et l'Optimisation par Essaims Particulaires.

De plus, les opérations qui créent de nouvelles solutions à partir de celles existantes ont un impact important sur la vitesse de convergence et la diversité des populations [Smith, 2004]. Les algorithmes d'optimisation qui favorisent l'intensification au détriment de la diversification ont une plus grande vitesse de convergence, mais courent le risque de ne pas trouver la solution optimale et peut rester coincé dans un optimum local. Puis à nouveau, les algorithmes qui effectuent trop de diversification peuvent ne jamais améliorer suffisamment leurs candidats solution pour trouver l'optimum global, ou il peut prendre très longtemps pour la découvrir «par accident». Un bon exemple de ce dilemme est l'algorithme de recuit simulé [Kirkpatrick S et al., 1983]. Il est souvent modifié sous une forme qui se concentre sur l'intensification, mais perd la convergence garantie à l'optimum [Ingber, 1996].

En règle générale, des algorithmes d'optimisation devraient employer au moins une opération de recherche de type exploratoire et au moins une qui soit en mesure d'exploiter des solutions encore bonnes. Ne pas préserver un équilibre entre diversification et intensification peut conduire à une convergence trop rapide vers des minima locaux (manque de diversification) ou à une exploration trop longue (manque d'intensification). Plusieurs travaux ont été réalisés dans ce domaine [Amor et Rettinger], [Deb, 2001], [Eiben et Schippers, 1998], [Eshelman et al., 1989], [Holland, 1992] et [Muttil et Liong, 2004]. Une discussion approfondie et assez complète sur l'intensification et la diversification peut être également trouvée dans [Blum et Roli, 2003].

Un grand nombre de métaheuristiques existent dans la littérature. N'étant pas, à priori, spécifiques à la résolution de tel ou tel type de problème, leur classification reste assez arbitraire. Différentes façons de les classer et les décrire sont proposées. La section suivante offre un aperçu des différentes approches existantes.

2.3 Classification des métaheuristiques

Les métaheuristiques constituent une classe d'algorithmes qui ne cesse de s'élargir. On peut citer : le Recuit Simulé (RS), la Recherche Tabou (RT), la Recherche Locale Itérée (RLI), Evolutionary Computation (EC), Ant Colony Optimization (ACO), Scatter Search (SS) , Greedy Randomized Adaptive Search Procedures (GRASP) et Algorithme à Estimation de Distribution.

Une façon intuitive de classer les métaheuristiques consiste à séparer celles qui sont inspirées d'un phénomène naturel de celles qui ne le sont pas. Les Algorithmes Evolutionnaires, très largement utilisés, appartiennent à la classe des métaheuristiques inspirées de la nature. D'autres techniques de popularité croissante dans cette classe comprennent l'optimisation par colonies de fourmis, l'optimisation par essaim de particules, les systèmes immunitaires artificiels, et ainsi de suite. Scatter Search, Recherche tabou, ou Méthode de descente sont des exemples de métaheuristiques non inspirées de la nature. Cependant, il est parfois difficile de différencier ente les classes (par exemple, dans le cas de la Recherche Taboue, l'utilisation de la mémoire pourrait la faire classer dans la première catégorie).

On peut également raisonner par rapport à l'usage que font les métaheuristiques de la fonction objectif. Certaines la laissent inchangée d'un bout à l'autre du processus de résolution, tandis que d'autres la modifient en fonction des informations collectées au cours de l'exploration – l'idée étant toujours de « s'échapper » d'un minimum local, pour avoir davantage de chance de trouver l'optimum. La recherche guidée locale est un exemple de métaheuristique qui modifie la fonction objectif.

Une autre classification consiste à distinguer les métaheuristiques qui ont la faculté de mémoriser des informations à mesure que leur recherche avance, de celles qui fonctionnent sans mémoire, en aveugle, et qui peuvent revenir sur des solutions qu'elles ont déjà examinées. Le meilleur représentant des métaheuristiques avec mémoire reste la recherche Tabou. Dans les « sans mémoire », on trouve le recuit simulé par exemple.

Une autre approche de classification souvent rencontrée dans la littérature est la classification : population vs solution unique. Elle consiste à distinguer les méthodes à base de population de solutions des méthodes à base de solution courante unique. Les premières citées travaillent sur un ensemble de points de l'espace de recherche, comme les algorithmes évolutionnaires, les colonies de fourmis, l'optimisation par essaim particulaire, les algorithmes à estimation de distribution. Quant aux

méthodes à base de solution courante unique, elles travaillent sur un seul point de l'espace de recherche à un instant donné, comme les méthodes de recuit simulé, la recherche tabou.

Des Modèles unifiés de procédures d'optimisation par les métaheuristiques ont été proposés [Vaessens et al., 1992], [Vaessens et al., 1998], [Rayward-Smith, 1994], [Osman, 1995] et [Taillard et al].

Nous présentons ci-dessous, des métaheuristiques largement utilisées dans la pratique, en respectant l'approche de classification : métaheuristiques à solution courante unique vs métaheuristiques à population. Un intérêt particulier sera porté aux méthodes que nous avons utilisées dans nos travaux présentés dans les chapitres 2 et 3.

2.4 Métaheuristiques à solution courante unique

Les méthodes itératives à solution courante unique sont toutes basées sur un algorithme de recherche qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage [Bachelet, 1999]. Cette idée vient du fait qu'une bonne stratégie pour la résolution des problèmes d'optimisation consisterait à se déplacer à travers l'espace de recherche en effectuant de petits pas (petits changements dans la solution courante) dans des directions qui améliorent la fonction objectif. Ces méthodes sont appelées méthodes de recherche locale, dont la version la plus simple est la méthode de descente.

2.4.1 Méthodes de descente (Hill Climbing)

La méthode de descente procède de manière itérative, en choisissant à chaque itération un point dans le voisinage de la solution courante. S'il est meilleur, il devient la nouvelle solution courante, sinon un autre point est choisi, et ainsi de suite. Une excellente introduction à cette méthode peut être trouvée dans [Papadimitriou et Steiglitz, 1982].

Algorithme 1 : Pseudo code de la Méthode de descente générique

Déterminer une configuration aléatoire s_0
Tant que le critère d'arrêt n'est pas atteint **faire**
  Générer *N(s)* ; /* Génération des candidats voisins*/
  Si pour tout $s' \in N(s)$ $f(s') \leq f(s)$ **Alors** Arrêt ;
  s := s' ; /* Sélection d'un meilleur voisin $s' \in N(s)$ */
Fin Tant que.
Résultat : Solution finale (optimum local)

On distingue plusieurs types de descente en fonction de la stratégie de génération de la solution de départ et du parcours du voisinage : la descente déterministe, la descente aléatoire et la descente vers le premier meilleur.

a. descente aléatoire : la nouvelle solution est choisie aléatoirement,
b. descente déterministe : le meilleur voisin est choisi,
c. descente vers le premier meilleur : le premier voisin meilleur est choisi.

En général, l'efficacité des méthodes de recherche locale simples (descente, ou plus grande descente) est très peu satisfaisante en général. D'abord, par définition, la recherche s'arrête au premier minimum local rencontré, c'est là leur principal défaut. Pour améliorer les résultats, on peut lancer plusieurs fois l'algorithme en partant d'un jeu de solutions initiales différentes, (recherche locale répétée), mais la performance de cette technique décroît rapidement. En revanche, autoriser de temps à autre une certaine dégradation des solutions trouvées, pour mieux explorer l'espace des configurations, a conduit au développement des deux méthodes que sont la Recherche Tabou et le Recuit Simulé.

Le principal avantage de la recherche locale simple est évidemment sa grande simplicité de mise en œuvre: la plupart du temps, elle ne fait que calculer f(s')-f(s).

Il est important de remarquer également l'importance du choix de la fonction de voisinage N : un minimum local pour une certaine structure de voisinage ne l'est pas forcément pour une autre. C'est d'ailleurs ce constat qui est à l'origine de la méthode dite de recherche par voisinage variable, qui repose sur la construction de solutions s parmi plusieurs voisinages, plutôt que dans un seul.

2.4.2 Le Recuit Simulé

La méthode du recuit simulé [Kirkpatrick S et al., 1983] a été introduite en 1983. Elle s'inspire d'une procédure utilisée depuis longtemps par les métallurgistes qui, pour obtenir un alliage sans défaut, chauffent d'abord à blanc leur morceau de métal, avant de laisser l'alliage se refroidir très lentement (technique du recuit). Pour simuler cette évolution d'un système physique vers son équilibre thermodynamique à une température *T*, la méthode du recuit simulé exploite l'algorithme de Métropolis [Metropolis, 1953].

Dans l'algorithme de Métropolis, on part d'une solution donnée, et on fait subir au système une modification élémentaire. Si cette perturbation a pour effet de diminuer la fonction objectif (ou *énergie*) du système, elle est acceptée. Sinon, elle est acceptée avec la probabilité $e^{-\Delta E/T}$. En appliquant itérativement cette règle, on engendre une séquence de solutions qui tendent vers l'équilibre thermodynamique.

Algorithme 2 : Pseudo-code de l'algorithme de Metropolis

Initialiser un point de départ x_0 et une température T
Pour i = 1 à n **faire**
  Tant que x_i n'est pas accepté **faire**
    Si $f(x_i) \leq f(x_{i-1})$: accepter x_i
    Si $f(xi) > f(x_{i-1})$: accepter x_i avec la probabilité $e^{-(f(x_i) - f(x_{i-1}))/T}$
  Fin Tant que
Fin Pour

Algorithme 3 : pseudo-code de l'algorithme de recuit simulé

Déterminer une configuration aléatoire s
Choix des mécanismes de perturbation d'une configuration
Initialiser la température T
Tant que la condition d'arrêt n'est pas atteinte faire
  Tant que l'équilibre n'est pas atteint faire
    Tirer une nouvelle configuration S'
    Appliquer la règle de Metropolis
    Si f(s') < f(s)
      s_{min} = s'
      f_{min} = f(s')
    Fin Si
  Fin Tant que
Décroître la température
Fin Tant que

L'interprétation du fonctionnement de l'algorithme de recuit simulé est la suivante :

Soir r un nombre aléatoire généré dans [0, 1]

· Si *f(s')<f(s)* alors $e^{\frac{f(s)-f(s')}{T}}$ >1, donc r est toujours inférieur à cette valeur, et on accepte la solution *s'* (une meilleure solution est donc toujours acceptée, ce qui paraît logique).

· Si *f(s')>f(s)* et *T* est très grand, alors $e^{\frac{f(s)-f(s')}{T}}$ ≈1, et il y a de fortes chances d'accepter *s'* (bien que la solution *s'* soit plus « mauvaise » que *s* !)

· Si *f(s')>f(s)* et *T* est très petit, alors $e^{\frac{f(s)-f(s')}{T}}$ ≈0, et on va donc probablement refuser *s'*

Dans un premier temps, *T* étant généralement choisi très grand, beaucoup de solutions, même celles dégradant la valeur de *f*, sont acceptées, et l'algorithme équivaut à une visite aléatoire de l'espace des configurations. Mais à mesure que la température baisse, la plupart des mouvements augmentant l'énergie sont refusés, et l'algorithme se ramène à une amélioration itérative classique. A température

intermédiaire, l'algorithme autorise de temps en temps des transformations qui dégradent la fonction objectif. Il laisse ainsi une chance au système de s'extraire d'un minimum local.

Le point crucial de l'algorithme est la loi de décroissance de la température. De nombreuses lois de décroissance différentes ont été proposées [Triki et al., 2005]. On fait souvent suivre à *T* une loi géométrique décroissante : $T_{k+1} = T_k \cdot \alpha$, avec α = 0.9 par exemple. *T* peut décroître linéairement, à chaque itération, mais on peut aussi envisager une décroissance par paliers, c'est-à-dire en gardant *T* constant, tant qu'un certain nombre de conditions n'ont pas été remplies, de façon à atteindre l'équilibre thermodynamique.

Une fois cet équilibre atteint, on abaisse légèrement la température et on recommence une nouvelle chaîne de calculs à ce nouveau palier. Dans la pratique, le processus est stoppé lorsque le système s'est figé, c'est à dire lorsque la température a atteint la valeur nulle ou bien lorsque plus aucun mouvement accroissant l'énergie n'a été accepté au cours du palier : par exemple, lorsque trois paliers successifs de température ont été descendus sans qu'aucune solution nouvelle n'ait pu être trouvée.

L'algorithme de recuit simulé est devenu rapidement populaire, du fait de sa facilité d'adaptation à un grand nombre de problèmes et de son efficacité. Aarts, Korst et Laarhoven [Aarts et al., 1997] ont par ailleurs démontré que, sous certaines conditions de décroissance de la température, l'algorithme du recuit simulé converge en probabilité vers un optimum global lorsque le nombre d'itérations tend vers l'infini.

L'un des inconvénients du recuit simulé est qu'une fois l'algorithme piégé à basse température dans un minimum local, il lui est impossible de s'en sortir tout seul. Plusieurs solutions ont été proposées pour tenter de résoudre ce problème, par exemple en acceptant une brusque remontée de la température, de temps en temps, pour relancer la recherche sur d'autres régions plus éloignées. Il est également possible d'empêcher la température de descendre trop bas : on lui donne une valeur minimale au delà de laquelle on ne change plus de palier de température. Mais si cette valeur est trop grande, l'algorithme passera son temps à augmenter et diminuer son énergie car il acceptera trop de perturbations dégradantes et il n'arrivera pas à explorer à fond une vallée.

Ainsi, il est fort possible que l'algorithme arrive à "trouver" la vallée dans laquelle se cache un minimum global, mais il aura beaucoup de mal à l'explorer et donc risque de s'en éloigner sans avoir décelé la solution au problème…

De plus, cet algorithme présente l'inconvénient de disposer d'un nombre élevé de paramètres (température initiale, règle de décroissance de la température, durée des paliers de température, etc.) qui rendent les réglages de l'algorithme assez empiriques.

2.4.3 La recherche Tabou

La recherche Tabou est une métaheuristique développée par Glover [Glover et Laguna, 1997]. Cette approche combine une procédure de recherche locale avec un ensemble de règles lui permettant de surmonter l'obstacle des extremums locaux ainsi que les problèmes de cycles.

Dans cette recherche locale, la meilleure solution dans le voisinage est sélectionnée comme nouvelle solution courante, même si elle n'améliore pas la solution actuelle. Cette politique peut générer des cycles, c'est à dire des solutions choisies précédemment qui pourraient être à nouveau sélectionnées. Pour résoudre ce problème, l'algorithme utilise une mémoire permettant de conserver la trace de dernières meilleures solutions déjà rencontrées.

Algorithme 4 : Pseudo-code de l'algorithme de Recherche Tabou

Déterminer une configuration aléatoire s
Initialiser une liste tabou vide
Tant que le critère d'arrêt n'est pas atteint **faire**
  Perturbation de s suivant N mouvements non tabous
  Évaluation des N voisins
  Sélection du meilleur voisin t
  Actualisation de la meilleure position connue s*
  Insertion du mouvement t ⟶ s dans la liste tabou
  s = t
Fin Tant que

Il s'agit de solutions considérées comme tabou, elles sont regroupées dans une liste tabou. Une nouvelle solution sera acceptée seulement si elle n'appartient pas à cette liste. Ce critère d'acceptation d'une nouvelle solution évite le bouclage de l'algorithme sur des solutions déjà enregistrées dans la liste tabou et il dirige l'exploration de l'espace des solutions vers des zones encore non visitées.

L'algorithme de recherche tabou peut être résumé par l'Algorithme 4. Dans sa forme de base, l'algorithme de recherche tabou présente l'avantage de comporter moins de paramètres que l'algorithme de recuit simulé. Cependant, l'algorithme n'étant pas toujours performant, il est souvent approprié de lui ajouter des processus d'intensification et/ou de diversification, qui introduisent de nouveaux paramètres de contrôle.

La méthode Tabou est une méthode de recherche locale, avec la particularité d'avoir un paramétrage simplifié : dans un première temps, le paramétrage consistera d'abord à trouver une valeur indicative *t* du nombre d'itérations pendant lesquelles les mouvements sont interdits. Il faudra également décider d'une stratégie de mémorisation à long terme – sur la qualité des solutions, sur leur ancienneté, …

En revanche, la méthode Tabou exige une gestion de la mémoire de plus en plus lourde à mesure que l'on voudra raffiner le procédé en mettant en place des stratégies de mémorisation complexe.

L'efficacité de la méthode Tabou fait qu'elle est largement employée dans les problèmes d'optimisation combinatoire : elle a été testée avec succès sur les grands problèmes classiques (voyageur de commerce, ordonnancement d'ateliers) et elle est fréquemment appliquée sur les problèmes de constitution de planning, de routage, d'exploration géologique, etc.

2.4.4 GRASP

GRASP (Greedy Randomized Adaptive Search Procedure ou Procédure de recherche gloutonne adaptative et aléatoire) est une autre méthode basée sur la recherche locale répétée, dans laquelle chaque recherche locale est initialisée avec une nouvelle solution réalisable construite grâce à une procédure de construction gloutonne et aléatoire [Feo et Resende, 1989, 1995]. On retrouve donc 2 phases principales à chaque itération : la construction des solutions de départ, et une recherche locale. En effet, l'algorithme construit une solution de départ, puis il effectue une recherche locale sur le voisinage de cette solution pour l'optimiser localement. Lorsque cela est fait, si la solution est meilleure que celle stockée précédemment, il la remplace et construit une nouvelle solution de départ aléatoirement. Un pseudo-code de l'algorithme de GRASP suit.

Algorithme 5 : Pseudo-code de l'algorithme GRASP

Déterminer une configuration aléatoire s
Tant que le critère d'arrêt n'est pas atteint **faire**
    Construction Aléatoire Gloutonne d'une solution de départ
    Recherche Locale sur le voisinage de cette solution pour l'optimiser localement
        Actualisation de la meilleure solution connue
Fin Tant que

La méthode GRASP est relativement simple à programmer, et permet d'améliorer les résultats d'une recherche locale simple. Elle permet également de faire intervenir une heuristique spécialisée. Elle ne nécessite que peu de paramétrage : la taille de la liste, qui permet d'équilibrer la quantité d'adaptativité (l'heuristique) et le facteur stochastique. De plus, elle nécessite peu de calculs supplémentaires par rapport à une recherche locale simple.

Elle est en revanche moins performante que d'autres métaheuristiques, essentiellement du fait de son absence de mémoire : comme avec une recherche locale simple, rien ne garantit que l'algorithme ne va pas, à plusieurs reprises, explorer des zones très similaires et retomber sur les mêmes minima locaux.

D’autres exemples populaires de métaheuristiques à solution unique peuvent être trouvés dans [Talbi, 2009]. Nous citons :

- La recherche à voisinage variable [Mladenovic et Hansen, 1997] dont l’idée de base est de changer de structure de voisinage pour sortir d’un optimum local.
- La recherche locale guidée [Voudouris et Tsang, 1995] qui repose sur une modification dynamique de la fonction objectif, dans le but de pénaliser l’optimum local courant. L’idée est de répéter des recherches locales en modifiant le coût des éléments constitutifs d’une solution en les pénalisant d’autant plus qu’ils sont fréquemment utilisés et que leur contribution à la dégradation de l’objectif est élevée.Les pénalités peuvent être vues comme une technique de diversification.
- La recherche locale réitérée [Lourenço et al., 2002] repose sur l’idée de base qui consiste à utiliser les optimums locaux trouvés au cours des recherches locales précédentes, pour construire le point initial qui sera utilisé pour démarrer la suivante.

2.5 Métaheuristiques à population de solutions

Les métaheuristiques à population de solutions améliorent, au fur et à mesure des itérations une population de solutions. L’intérêt de ces méthodes est d’utiliser la population comme facteur de diversité. Elles démarrent à partir d'une population initiale de solutions. Puis, elles appliquent itérativement la génération d'une nouvelle population et le remplacement de la population courante. Dans la phase de génération, une nouvelle population de solutions est créée. Dans la phase de remplacement, une sélection est effectuée à partir des populations courante et nouvelle. Ce processus se répète jusqu'à ce qu'un critère d'arrêt donné soit satisfait.

Nous pouvons distinguer dans cette classe de méthodes plusieurs sous classes dont :

- les algorithmes évolutionnaires
- les algorithmes de colonies de fourmis
- les méthodes à particules : Essaims particulaires
- les algorithmes à estimation de distributions
- La recherche par dispersion

Un examen de ces méthodes est proposé dans [Talbi, 2009]. Nous donnons dans ce qui suit une présentation générale de quelques méthodes. Un intérêt particulier est porté aux métaheuristiques utilisées dans nos travaux présentés dans les chapitres 2 et 3.

2.5.1 Algorithmes évolutionnaires

Le paradigme des algorithmes évolutionnistes [Baeck et al., 2000] fondé dans les années 1960, consiste à s'inspirer des mécanismes de l'évolution naturelle et à utiliser le concept de populations d'individus ou solutions pour résoudre des problèmes du monde réel. La mise en oeuvre relativement facile de ces algorithmes ainsi que les nombreux succès qu'elles ont obtenus ont contribué à leur développement spectaculaire ces vingt dernières années.

Un algorithme évolutionnaire est une méthode itérative qui applique des opérateurs stochastiques sur un groupe d'individus. Chaque individu dans la population est la version codée d'une solution provisoire. Initialement, cette population est habituellement générée de façon aléatoire. À chaque génération de l'algorithme, des solutions sont sélectionnées, jumelées et recombinées afin de générer de nouvelles solutions qui remplacent les plus mauvaises selon certains critères, et ainsi de suite. Une fonction associe une valeur à chacun des individus indiquant son aptitude au problème (critère de sélection).

Les approches évolutionnaires s'appuient toutes sur un modèle commun présenté par l'Algorithme 6. Au cours de chaque génération, une succession d'opérateurs est appliquée aux individus de la population pour engendrer une nouvelle population, en vue de la génération suivante. Chaque opérateur utilise un ou plusieurs individus de la population appelé(s) parent(s) pour engendrer de nouveaux individus, appelés enfants. A la fin de chaque génération, un certain nombre d'individus de la population sont remplacés par des enfants créés durant la génération.

Un algorithme évolutionnaire dispose donc de trois opérateurs principaux : un opérateur de sélection, un opérateur de croisement et un opérateur de mutation. L'opérateur de sélection permet de favoriser la propagation des meilleures solutions dans la population, tout en maintenant la diversité génétique de celle-ci.

Algorithme 6 : Pseudo-code de l'algorithme évolutionnaire générique

Initialisation de la population de μ individus
Évaluation des μ individus
Tant que le critère d'arrêt n'est pas atteint **faire**
  Sélection de λ individus en vue de la phase de reproduction
  Croisement des λ individus sélectionnés
  Mutation des λ individus sélectionnés
  Évaluation des λ' enfants obtenus
  Sélection pour le remplacement
Fin Tant que

L'opérateur de croisement est mis en oeuvre lors de la phase de création des enfants pour échanger les gènes des différents parents et créer les enfants (figure 2.1).

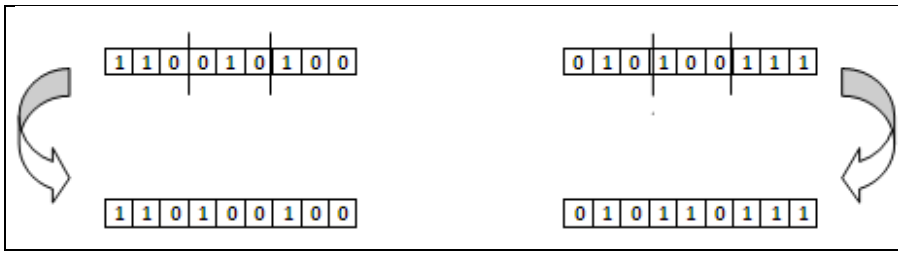


Figure 2.1 - Exemple d'opérateur de croisement simple
(Individu codé en représentation binaire).

L'opérateur de mutation consiste à tirer aléatoirement une composante de l'individu parent et à la remplacer par une valeur aléatoire. L'opérateur de mutation apporte un caractère aléatoire à la création de la descendance, et permet ainsi de maintenir une certaine diversité dans la population (Figure 2.2).

Une liste de méthodes existantes pour définir ces opérateurs est disponible dans [Eiben et Smith, 2003].

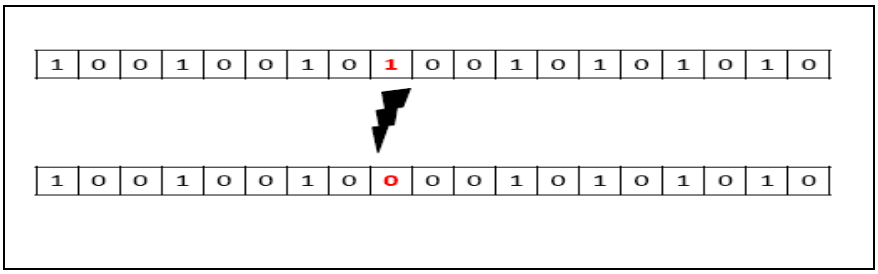


Figure 2.2 - Exemple d'opérateur de mutation
(Individu codé en représentation binaire).

Il existe plusieurs sous-classes d'algorithmes évolutionnaires parmi lesquelles les algorithmes génétiques, les méthodes de programmation évolutionnaire développées par Fogej [Back et al., 2000], la programmation génétique, les stratégies évolutionnaires développées indépendamment par Rechenberg(Rec73) et Schwefel(Sch81), etc. Une description détaillée est faite dans [Back et al., 1997]. Nous présentons ci-dessous le déroulement général des algorithmes génétiques, ainsi que des métaheuristiques « Biogeography Based Optimisation » (BBO) et « Harmony Search » (HS) que nous utilisons au chapitre 2.

2.5.1.1 Algorithmes Génétiques

Proposés dans les années 1975 par Holland [Holland, 1975], [Holland, 1992], les algorithmes génétiques doivent leur popularité à Goldberg [Goldberg, 1989]. Ils constituent des méthodes basées sur les mécanismes biologiques tels que les lois de Mendel et sur le principe fondamental (selection) de Charles Darwin [Darwin, 1859]. Les algorithmes génétiques simulent le processus d'évolution d'une population. Ils s'appuient sur trois fonctionnalités :

- La sélection qui permet de favoriser les individus qui ont un meilleur fitness (valeur de la fonction objectif de la solution associée à l'individu).
- Le croisement qui combine deux solutions parents pour former un ou deux enfants (offspring) en essayant de conserver les "bonnes" caractéristiques des solutions parents.
- La mutation qui permet d'ajouter de la diversité à la population en mutant certaines caractéristiques (gènes) d'une solution.

D'une manière générale, et à partir d'une population de N solutions du problème représentant des individus, on applique des opérateurs simulant les interventions sur le génome (croisement, mutation) pour arriver à une population de solutions de mieux en mieux adaptée au problème. Cette adaptation est évaluée grâce à une fonction coût (fitness). Pour utiliser un algorithme génétique, on doit disposer des cinq éléments suivants :

- La représentation des solutions ou le codage. La qualité du codage des données conditionne le succès des algorithmes génétiques comme précisé dans la section 2 du présent chapitre. Il doit s'adapter le mieux possible au problème et à l'évaluation d'une solution. Cette étape associe à chacun des points de l'espace d'état une structure de données.
- Un mécanisme de génération de la population initiale. Il doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global (problème de la convergence prématurée vue dans le chapitre précédent).
- Une fonction à optimiser. Celle-ci retourne une valeur de R^+ appelée *fitness* ou fonction coût (d'évaluation) de l'individu.
- 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 et l'opérateur de mutation a pour but de garantir l'exploration de l'espace des états.
- 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.

Ainsi, dans le fonctionnement général d'un algorithme génétique, on commence par générer une population d'individus de façon aléatoire. Pour passer d'une génération k à la génération k+1, les trois opérations suivantes sont répétées pour tous les éléments de la population k. Des couples de parents P_1 et P_2 sont sélectionnés en fonction de leurs adaptations. L'opérateur de croisement leur est appliqué avec une probabilité p et génère des couples d'enfants C_1 et C_2. D'autres éléments P sont sélectionnés en fonction de leur adaptation. L'opérateur de mutation leur est appliqué avec la probabilité p_m (p_m est généralement très inférieur à p) et génère des individus mutés P'. Le niveau d'adaptation des enfants (C_1, C_2) et des individus mutés P' sont ensuite évalués avant insertion dans la nouvelle population. Différents critères d'arrêt de l'algorithme peuvent être choisis tels que le nombre de générations que l'on souhaite exécuter ou lorsque la population n'évolue plus ou plus suffisamment rapidement.

Les algorithmes génétiques connaissent une grande popularité. Un très grand nombre de références dédiées au domaine existe. On citera les livres [Haupt et Haupt, 1998] et [Mitchel, 1998]. Cerf apporte la preuve de convergence de la méthode dans [Cerf, 1994].

On notera que les algorithmes génétiques sont coûteux en temps de calcul, puisqu'ils manipulent plusieurs solutions simultanément. C'est le calcul de la fonction de performance qui est le plus pénalisant, et on optimise généralement l'algorithme de façon à éviter d'évaluer trop souvent cette fonction. Ensuite, l'ajustement d'un algorithme génétique est délicat. L'un des problèmes les plus caractéristiques est celui de la dérive génétique, qui fait qu'un bon individu se met, en l'espace de quelques générations, à envahir toute la population. On parle dans ce cas de convergence prématurée, qui revient à lancer à une recherche locale autour d'un minimum… qui n'est pas forcément l'optimum attendu. Un autre problème surgit lorsque les différents individus se mettent à avoir des performances similaires : les bons éléments ne sont alors plus sélectionnés, et l'algorithme ne progresse plus.

Le grand avantage des algorithmes génétiques est qu'ils parviennent à trouver de bonnes solutions sur des problèmes très complexes. Ils doivent simplement déterminer entre deux solutions quelle est la meilleure, afin d'opérer leurs sélections. Par ailleurs, les algorithmes génétiques se prêtent bien, du fait de leur traitement simultané de solutions, à la recherche d'optimum multiples : en créant une fonction de coût partagée, dont la valeur dépend partiellement de la distance entre les individus, on voit se former graduellement des sous-populations d'individus, qui se stabilisent autour des différents pics de la fonction objectif. C'est la technique du nichage par la méthode du partage.

2.5.1.2 Biogeography Based Optimization (BBO)

2.5.1.2.1 Introduction

Les modèles mathématiques de la biogéographie décrivent la migration, la spéciation, et l'extinction des espèces. La science de la biogéographie a commencé avec des descriptions empiriques par des naturalistes du 19ème siècle tels que Alfred Wallace [Wallace, 2005] et Charles Darwin

[Darwin, 1995].Robert Mac Arthur et Edward Wilson ont été les premiers à développer et largement les faire connaître dans les années 1960 [MacArthur et Wilson, 1967].

Les îles qui sont bien adaptées comme habitats pour les espèces biologiques sont dites avoir un indice d'habitabilité élevé. On note cet indice HSI (en anglais: Habitat Suitability Index). Les caractéristiques des îles qui sont corrélées avec HSI incluent les précipitations, la div4ersité, la topographie, la région, la température, etc. Les variables qui caractérisent ces caractéristiques sont notées SIV (en anglais Suitability Index Variable). Les SIV sont des variables indépendantes, et HSI est la variable dépendante.

Les îles avec un HSI élevé ont tendance à avoir un grand nombre d'espèces ; celles ayant un faible HSI ont un petit nombre d'espèces. Les Îles avec un HSI élevé ont de nombreuses espèces qui émigrent vers les îles voisines en raison de problèmes liés à la surpopulation (sur des épaves, en volant ou à la nage).

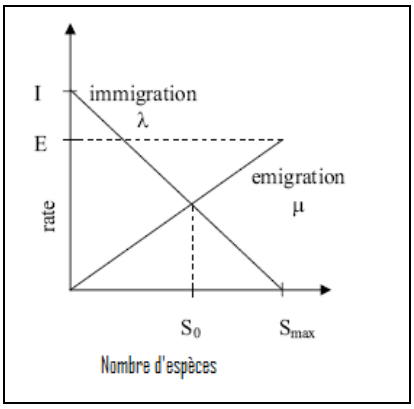


Figure 2.3 - Relations migratoires linéaires pour une île

Considérons le modèle particulier de l'abondance des espèces dans une seule île (Figure 2.3, [Simon, 2008]). C'est un modèle simple qui donne une description générale du processus d'immigration et d'émigration. Le taux d'émigration μ hors de l'île, et le taux d'immigration λ dans l'île, sont des fonctions (ici linéaires) du nombre d'espèces S sur l'île. Le taux d'immigration maximum possible I se produit lorsque le nombre d'espèces sur l'île est nul. Quand le nombre d'espèces augmente, l'île devient plus encombrée, le taux d'immigration diminue. Le plus grand nombre possible d'espèces que l'île peut contenir est S_{max} correspondant au point où le taux d'immigration est égal à zéro. S'il n'y a pas d'espèces sur l'île, alors le taux d'émigration est zéro. Comme le nombre d'espèces augmente, l'île devient de plus en plus peuplée, des individus représentatifs des espèces sont plus susceptibles de quitter l'île, et le taux d'émigration augmente. Le taux d'émigration maximale E se produit lorsque l'île contient le plus grand nombre d'espèces qu'il peut prendre en charge.

Se basant sur les concepts de biogéographie exposés plus haut, D. Simon a développé en 2008 [Simon, 2008] une nouvelle approche de résolution de problèmes d'optimisation qui est venue enrichir la famille des algorithmes évolutionnaires. Il s'agit de Biogeography Based Optimization (BBO) dont nous donnons une présentation ci-dessous. Considérons un problème à résoudre avec des solutions candidates que nous pouvons évaluer. Dans son principe de résolution, BBO utilise les analogies suivantes :

• Une solution est analogue à un habitat

• La qualité de la solution(fitness)est analogue au HSI de l'habitat.

• Les variables définissant la solution sont analogues aux SIV.

• Une bonne solution est analogue à un habitat avec un HSI élevé, et donc possédant un grand nombre d'espèces, un taux d'émigration élevé et un taux d'immigration faible

• Une mauvaise solution est analogue à un habitat avec un HSI faible, et donc possédant un nombre d'espèces faible, un taux d'émigration faible et un taux d'immigration élevé.

• Une bonne solution tend à partager les caractéristiques d'une mauvaise solution pour l'améliorer (la migration des SIV). Ceci est analogue à la migration des espèces entre les habitats. Partager des caractéristiques n'implique pas le changement dans les caractéristiques des bonnes solutions.

• Les mauvaises solutions tendent à accepter les caractéristiques des bonnes solutions afin d'améliorer leur qualité. Ceci est analogue à un mauvais habitat qui accepte l'immigration des espèces d'autres habitats.

Ainsi, si en biogéographie, les espèces migrent entre les îles, dans BBO ce sont les caractéristiques (SIV) de la solution qui migrent. Nous utilisons les taux de migration de chaque solution pour partager probabilistiquement des caractéristiques entre les solutions. Ceci peut être mis en œuvre de plusieurs manières différentes. Dans la formulation d'origine de BBO, appelée « Partial Immigration Based BBO » [Simon, 2009], pour chacune des caractéristiques de chaque solution, la courbe d'immigration (pour des raisons de simplicité, nous supposons que toutes les solutions – îles- ont des courbes de migration identiques avec E = I) est utilisée pour décider probabilistiquement si il y'a immigration ou non. Si l'immigration est sélectionnée pour une caractéristique d'une solution donnée, l'île d'émigration est alors sélectionnée de manière probabiliste (par exemple en utilisant la roulette).

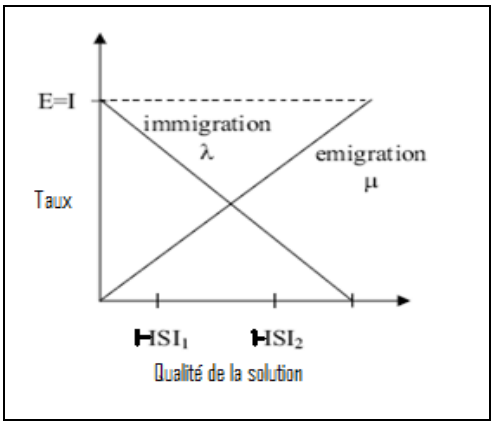


Figure 2.4 - Illustration de deux solutions candidates à un problème à l'aide de courbes d'immigration et d'émigration symétriques.

La Figure 2.4 illustre les courbes de migration ainsi que deux solutions. HSI_1 représente une mauvaise solution et HSI_2 représente une bonne solution. Le taux d'immigration pour ISI_1sera donc plus élevé que le taux d'immigration pour ISI_2, et le taux d'émigration pour HSI_1 sera inférieur au taux d'émigration de HSI_2.

2.5.1.2.2 Migration

Faire évoluer la population (améliorer la qualité des individus de la population) est le moyen utilisé par les heuristiques à population pour résoudre les problèmes. Dans BBO, l'itération pour la prochaine génération se fait par l'immigration des caractéristiques d'une solution vers d'autres habitats, et par la réception de caractéristiques d'une solution par émigration à partir d'autre habitats.

Algorithme 7: Pseudo-code de l'algorithme Migration

```
Sélectionner Hi avec la probabilité αλi
  Si Hi sélectionné
    Pour j=1 à n faire
      Sélectionner Hj avec une probabilité αμj
        Si Hj selectionné
          Remplacer le SIV dans Hi par le SIV de HJ
        Fin Si
    Fin Pour
  Fin Si
```

Supposons connue une population de solutions candidates à un problème, représentées par des vecteurs (Habitats H_i, i=1…n). Chacune des composantes du vecteur est considérée comme la valeur d'une SIV. Dans le processus de migration (algorithme 7 [Simon 2008]), quand une solution est choisie pour modification (avec une probabilité P_{mod} : une valeur fixée par l'utilisateur), le taux d'immigration $\lambda(1)$ est utilisé pour décider si une SIV doit être modifiée. Puis, le taux d'émigration μ (2) est utilisé pour décider laquelle des bonnes solutions migrera son SIV.

Dans le cas de la Figure 1.8, nous avons :

$$\lambda_k = I(1 - \frac{k}{N}) . \quad (1)$$

$$\mu_k = E * \frac{K}{N} \quad . \quad (2)$$

où:

- λ_k est le taux d'immigration d'un habitat avec *k* espèces
- μ_k est le taux d'émigration dans un habitat avec *k* espèces
- E est le taux maximum d'émigration
- I est le taux maximum d'immigration
- N est le nombre maximum d'espèces

2.5.1.2.3 Mutation

Chaque solution a aussi (comme dans tout algorithme évolutionnaire) généralement une certaine probabilité de mutation, même si la mutation n'est pas une caractéristique essentielle de BBO . Un cataclysme peut changer radicalement le HSI d'un habitat naturel. Elles peuvent également modifier le nombre d'une espèce de sa valeur d'équilibre (nombre d'épaves exceptionnellement élevé en provenance d'un habitat voisin, la maladie, les catastrophes naturelles, etc). Le HSI d'un habitat peut, par conséquent, changer brusquement à cause, apparemment, d'événements aléatoires. Ceci a été modélisé dans BBO par la mutation des SIV.

La distribution du nombre d'espèces est utilisée pour la détermination des taux de mutation suivant :

$$m(s) = \alpha \frac{1-P_s}{P_{max}} . \quad (3)$$

où:

- m(s) : taux de mutation d'un habitat avec s espèces.
- α : paramètre défini par l'utilisateur
- P_s : probabilité d'avoir s espèces dans l' habitat (4)
- P_{max} : probabilité d'avoir le maximum d'espèces

et :

$$P_k = \begin{cases} \dfrac{\lambda_0\lambda_1..\lambda_{k-1}}{\mu_1\mu_2..\mu_k(1+\sum_{l=1}^{n}\frac{\lambda_0\lambda_1...\lambda_{l-1}}{\mu_1\mu_2...\mu_l})} & 1 \le k \le n \\ \dfrac{1}{(1+\sum_{l=1}^{n}\frac{\lambda_0\lambda_1..\lambda_{l-1}}{\mu_1\mu_2..\mu_l})} & k = 0 \end{cases} \quad (4)$$

Remarque: les habitats mutés ont un HSI faible. Cette mutation introduit la diversité et encourage les habitats pauvres à s'améliorer.

Algorithme 8: Pseudo-code de l'algorithme de Mutation

Pour j=1 à m **faire**
  Calculer P_i en utilisant λ_i et μ_i
  Sélectionner SIV dans $H_i(j)$ avec la probabilité αP_i
  Si $H_i(J)$ sélectionné
    Remplacer $H_i(j)$ par un SIV choisi aléatoirement
  Fin Si
Fin Pour

La procédure dans l'algorithme Partial Immigration-based BBO ([Simon, 2009]) est donnée ci-dessous, où y est la population entière de solutions, y_k est la solution k, et $y_k(s)$ est la caractéristique s de y_k (individu k de la population).

Algorithme 9 : Pseudo-code Partial Immigration-based BBO

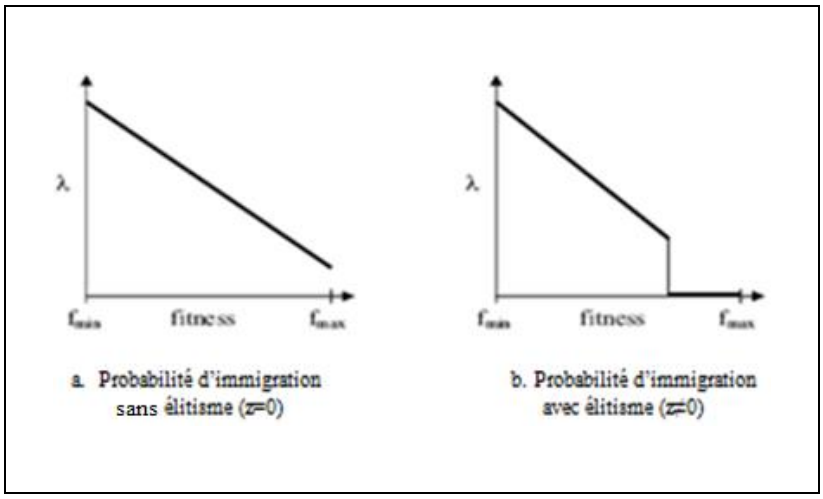
Noter y la population de solutions
w = y (w est un vecteur population temporaire)
Pour toute ile w_k
  Pour tout SIV s **faire**
    Immigrer vers w_k (s) avec la probabilité λ_k
    Si Immigration
      Sélectionner l'île d'émigration y_j avec la probabilité μ
      w_k (s) = y_j (s)
    Fin Si
    Décider probabilistiquement si mutation de w_k (s)
  Fin Pour
Fin Pour
y=w

La migration et la mutation de l'ensemble de la population aura lieu avant que toute solution soit remplacée dans la population, ce qui nécessite l'utilisation du vecteur temporaire de la population w.

Dans les stratégies évolutionnaires, la recombinaison globale est utilisée pour créer des solutions nouvelles, tandis que la migration est utilisée pour changer les solutions existantes dans BBO.

2.5.1.2.4 L'élitisme

Comme avec d'autres algorithmes d'optimisation basés sur la population, une sorte d'élitisme est habituellement incorporé dans le but de conserver les meilleures solutions dans la population. Cela empêche les meilleures solutions d'être corrompues par l'immigration. Un nombre z désigne le nombre de meilleurs individus dans la population ayant une probabilité nulle d'immigration. Si l'élitisme n'est pas utilisé, alors z=0 et λ la probabilité d'immigration est une fonction linéaire de la fitness et est positive pour toutes les valeurs de la fitness. Ceci est illustré dans la figure 2.5(a). Si l'élitisme est utilisé, alors z>0 et la probabilité d'immigration λ est nulle pour les z meilleurs individus de l'espace de recherche. Ceci est illustré dans la figure 2.5(b) [Simon, 2009].



a. Probabilité d'immigration sans élitisme (z=0)

b. Probabilité d'immigration avec élitisme (z=0)

Figure 2.5 - Probabilité d'immigration λ comme fonction de la fonction objectif

2.5.1.2.5 Comparaison de BBO à d'autres algorithmes évolutionnaires

BBO a certaines caractéristiques en commun avec d'autres algorithmes évolutionnaires. Comme les algorithmes génétiques (AG) [Goldberg, 1989], BBO présente une façon de partager l'information entre les solutions. Toutefois, les solutions GA meurent généralement à la fin de chaque génération, tandis que les solutions BBO survivent, même si leurs caractéristiques changent durant la progression de l'algorithme. Bien que BBO est un algorithme basé population de solutions, il n'implique pas la reproduction ou la génération d'enfants. Ce qui la distingue des stratégies de reproduction tels que les Algorithmes Génétiques. BBO diffère également de l'optimisation par colonie de fourmis (ACO), qui génère un nouvel ensemble de solutions à chaque itération [Dorigo et al., 2004], [Dorigo et al., 2002]. BBO, d'autre part, maintient l'ensemble de ses solutions d'une itération à l'autre, en s'appuyant sur la migration pour adapter probabilistiquement ces solutions.

BBO a plus de points en commun avec des algorithmes tels que l'optimisation par essaim de particules (PSO) [Eberhart et al., 2001],[Eberhart et Shi., 2004]. Dans PSO, les solutions sont maintenues d'une itération à l'autre, mais chaque solution est capable d'apprendre de ses voisins et de

s'adapter quand l'algorithme progresse. PSO représente chaque solution comme un point dans l'espace, et représente la variation au fil du temps de chaque solution en tant que vecteur de vitesse. Toutefois, les solutions PSO ne changent pas directement, c'est plutôt leurs vitesses qui changent, et cela permet indirectement des changements de la position (c.-à-solution). BBO peut être comparée à PSO en ce que des solutions BBO sont modifiées directement via la migration à partir d'autres solutions (îles). Ainsi, les solutions BBO partagent directement leurs attributs (SIV) avec d'autres solutions. C'est le caractère distinctif de chaque algorithme évolutionnaire qui détermine ses forces et faiblesses relatives.

2.5.1.3 Harmony Search

2.5.1.3.1 Introduction

Harmony Search (HS) [Geem et al., 2001] est une métaheuristique à population de solutions proposée en 2001 par Geem et al. (2001). Elle imite le processus de l'improvisation musicale appliquée par les musiciens, chaque musicien improvisant des sons sur son instrument pour obtenir une belle harmonie suivant des standards d'esthétiques musicales. L'objectif du processus est de parvenir à un état parfait de l'harmonie, tout comme le processus d'optimisation cherche à trouver une solution optimale globale évaluée par une fonction objectif. L'analogie entre improvisation musicale et optimization (Figure 2.6) peut se concevoir de la façon suivante :

1. Chaque musicien (saxophoniste, contrebassiste, guitariste, …) correspond à une variable de décision (x_1, x_2, x_3, …);

2. Le registre de l'instrument de musique (saxophone = {Do, Ré, Mi}; contrebasse = {Mi, Fa, Sol}, et la guitare = {Sol, La, Si}) correspond à la plage de chaque valeur de la variable (x_1 = {100, 200, 300}; x_2 ={300, 400, 500}, et x_3 = {500, 600, 700}).

3. L'harmonie musicale à un instant donné correspond au vecteur solution à une itération donnée : si le saxophoniste joue la note Ré, le contrebassiste la note Mi, et le guitariste la note Si, leurs notes forment ensemble une harmonie nouvelle (Ré, Mi, Si).

4. L'esthétique musicale correspond à la fonction objectif, et si la nouvelle harmonie est meilleure que l'harmonie existante, elle est retenue : la nouvelle solution (200, 300, 700) est retenue si elle est meilleure que l'harmonie existante dans les termes de la valeur de la fonction objectif. La qualité de l'harmonie est améliorée par la pratique, répétition après répétition.

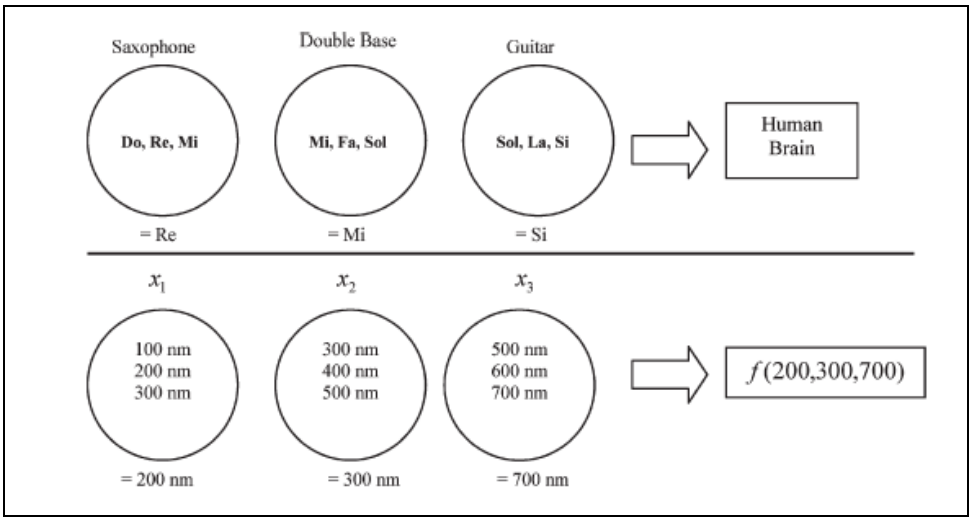


Figure 2.6 - Analogie entre improvisation musicale et optimisation

Tout comme l'harmonie musicale, la solution courante du problème d'optimisation s'améliore itération après itération.

2.5.1.3.2 Optimisation avec HS

Considérons le problème d'optimisation suivant:

Minimiser \ Maximiser f (x),
tel que: $x_i \in \mathbf{A}^i$, i = 1, 2, . . ., n (1)

Où : f (.) est la fonction objectif; $x = (x_1, \ldots, x_n)$ est le vecteur solution des variables de décision, $\mathbf{A}^i$ est l'ensemble des valeurs que peut prendre la variable de décision x_i. Dans le cas de variables discrètes, $x_i \in \{ x_i(1), \ldots, x_i(k), \ldots, x_i(K) \}$ et $x_i \in [{}_Lx_i \; {}_Ux_i]$ dans le cas de variables continues; n est le nombre de variables de décision.

L'application de HS nécessite préalablement le réglage des différents paramètres de control existant que nous expliciterons dans la suite. Il s'agit de :

- Harmony Memory Size (HMS) : nombre de vecteurs solution (nombre d'individus dans la population ou taille de la population) dans HM (à chaque génération)
- Harmony Memory Considering Rate (HMCR); HMCR ∈[0, 1]
- Pitch Adjusting Rate (PAR); PAR ∈[0, 1];
- Critère d'arrêt (NI) : nombre d'improvisations
- bw : band width

En général, l'algorithme HS comporte cinq étapes (Figure 2.7). Ainsi, et après l'étape d'initialisation du problème considéré (formulation et réglage des paramètres), le processus d'optimisation peut commencer.

- **Initialisation de la population**

Après que le problème soit formulé, et que les valeurs des paramètres soient correctement réglées, le processus d'initialisation aléatoire de la population est effectué.

En effet, comme dans un concert d'orchestre, après que le hautbois joue la note LA (généralement A440), les autres instruments jouent des notes au hasard de la gamme jouable. De même, l'algorithme HS génère initialement différentes solutions aléatoirement, en vérifiant (2). Soit HMS le nombre de ces vecteurs solutions générés. HMS est un paramètre fixé par l'utilisateur. Ensuite, les HMS vecteurs solutions de départ sont sauvegardées dans une matrice notée HM, de dimension (HMS x n), où chaque ligne représente un vecteur solution. HM (Harmony Memory) représente la population de solutions (en analogie avec la mémoire des musiciens).

Dans cette étape, les solutions sont donc construites aléatoirement, puis on les réordonne en fonction des valeurs de la fonction objectif ($f(x^1) \leq f(x^2) \ldots \leq f(x^{HMS})$).

$$HM = \begin{bmatrix} x_1^1 & x_2^1 \ldots & \ldots & x_n^1 & f(x^1) \\ x_1^2 & x_2^2 \ldots & \ldots & x_n^2 & f(x^2) \\ \vdots & & & & \\ x_1^{HMS} & x_2^{HMS} & \ldots & x_n^{HMS} & f(x^{HMS}) \end{bmatrix}$$

- **Génération d'une nouvelle solution**

Dans l'improvisation de jazz, un musicien joue une note aléatoirement sélectionnée, à partir de la gamme jouable de la mémoire du musicien, en ajustant la note. De même HS génère une solution en la choisissant parmi l'ensemble des valeurs dans HM et/ou en l'ajustant.

Ainsi, dans cette étape, HS génère un nouveau vecteur solution x'= ($x_1^{'}$,$x_2^{'}$,.., $x_n^{'}$) par application de trois opérateurs : Memory Consideration, Pitch Adjustment et Random Consideration.

Dans la phase Memory Consideration, les valeurs de chacune des composantes $x_i^{'}$ du nouveau vecteur solution x' sont choisies aléatoirement, avec la probabilité HMCR, HMCR ∈ [0, 1]., parmi ses valeurs précédentes dans HM. Aussi, la variable de décision $x_i^{'}$, pour tout i = 1… n, est choisie dans le vecteur (x_i^1, x_i^2, ,.., x_i^{HMS}) de HM, avec la probabilité HMCR ∈ [0, 1]. HMCR (en anglais: Harmony Memory Considering Rate) est un parameter fixé par l'utilisateur. L'indice j tel que $x_i^{'} = x_i^j$ peut être calculé en utilisant une distribution uniforme U(0, 1) : $j \rightarrow \text{int}(U(0, 1) * HMS) + 1$

Dans le cas où les autres valeurs de la variable de décision ne sont pas choisies dans HM, suivant le test de probabilité HMCR, elles sont choisies aléatoirement dans la gamme des valeurs possibles $x_i^{'} \in \mathbf{A}^i$ avec la probabilité (1−HMCR). Cette dernière phase, appelée : Random Consideration, améliore la diversité des solutions et conduit le système à explorer plus de solutions.

Les deux phases précédentes : Memory Consideration et Random Consideration, peuvent être résumées dans l'équation suivante:

$$x_i' \leftarrow \begin{cases} x_i' \in \left(x_i^1, x_i^2, , , \ldots, x_i^{HMS}\right) \text{avec une probabilité HMCR} \\ x_i' \in \mathbf{A}i \qquad \text{avec une probabilité: } (1 - \text{HMCR}) \end{cases}$$

Un ajustement est ensuite réalisé sur chaque variable de décision du nouveau vecteur solution : x'= $(x_1', x_2', .., x_n')$ avec la probabilité PAR de la façon suivante :

- Pour les variables discrètes, si $x_i' = x_i(k)$, la valeur ajustée devient :

$$x_i' \leftarrow \begin{cases} xi(k + m) & \text{avec probabilité PAR} \\ x_i' & \text{avec probabilité } (1 - \text{PAR}) \end{cases}$$

où généralement, m ∈ {-1, 1}.

- Pour les variables continues, la valeur ajustée devient

$$x_i' \leftarrow \begin{cases} x_i' = x_i' \pm rand(0\ 1) * \text{bw} & \text{avec probabilité PAR} \\ x_i' & \text{avec probabilité } (1 - \text{PAR}) \end{cases}$$

où PAR est un paramètre fixé par l'utilisateur, PAR ∈ [0, 1], bw (band witdh) est un scalaire et rand(0 1) est un nombre aléatoire généré suivant la loi uniforme dans [0, 1].

Ainsi si un nombre aléatoire rnd ∈ [0, 1] est généré dans la probabilité de PAR, alors, la nouvelle variable de décision x_i' sera ajustée.

En général, la façon dont le paramètre PAR modifie les composantes du nouveau vecteur solution est une analogie avec le comportement des musiciens quand ils changent légèrement leur fréquence de tonalité pour obtenir une meilleure harmonie. Par conséquent, il explore plus de solutions dans l'espace de recherche et améliore les capacités de recherche. Cette étape est responsable de la génération d'une nouvelle variation potentielle dans l'algorithme. Elle est comparable à la mutation dans les algorithmes évolutionnaires standards.

- **Mise à jour de HM**

Afin de mettre à jour HM avec le nouveau vecteur généré x'=$(x_1', x_2', .., x_n')$, la valeur de la fonction objectif f(x') de ce dernier est calculée. Si elle est meilleure que la plus mauvaise de toutes les valeurs des vecteurs solutions dans HM en termes de leur fonction objectif, alors le nouveau vecteur solution est inséré dans HM en remplacement du vecteur de plus mauvaise valeur de la fonction

objectif. Sinon le nouveau vecteur généré x'=(x_1', x_2',.., x_n') est ignoré. Toutefois, pour la diversité des harmonies dans HM, d'autres harmonies (en termes de moindre-similarité) peuvent être considérées. En outre, le nombre maximum d'harmonies identiques dans HM peut être pris en compte afin de prévenir la convergence prématurée.

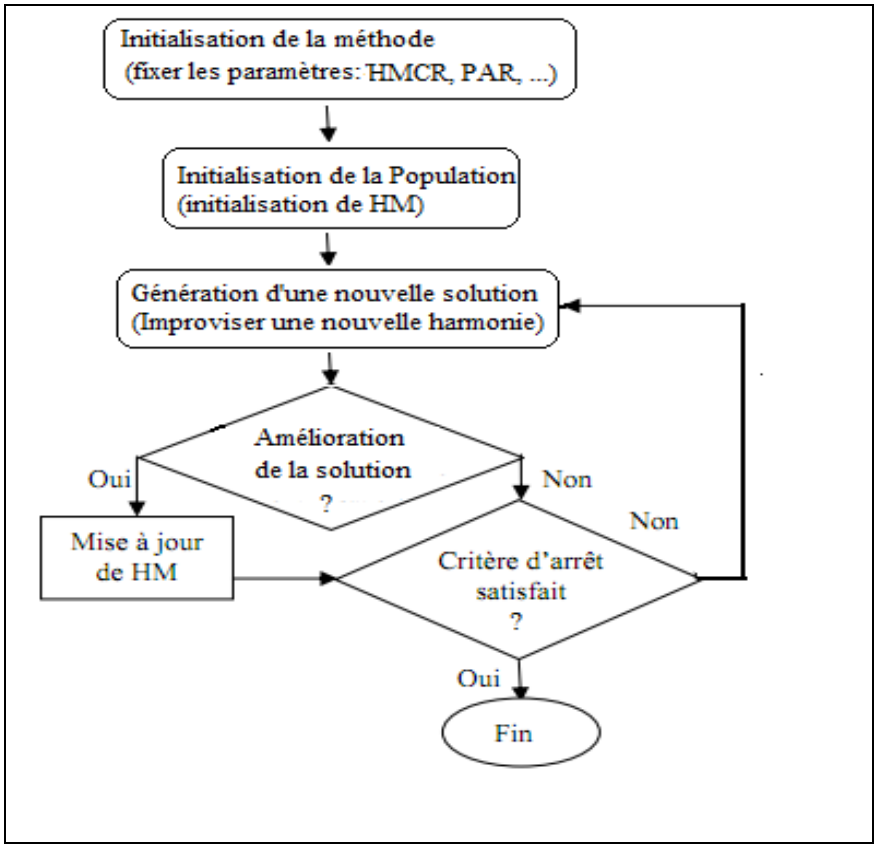


Figure 2.7 – Organigramme de la méthode Harmony Search

- **Critère d'arrêt**

Le processus d'itération dans les étapes 3 et 4 se termine quand le nombre d'improvisations NI est atteint. Enfin, le meilleur vecteur dans HM est sélectionné. Il est considéré comme la meilleure solution du problème étudié.

Harmony Search a été appliquée à la résolution de nombreux problèmes d'optimisation issus de domaines différents [Ingram et Zhang, 2009]. Nous pouvons citer entre autres : la composition musicale [Geem and Choi, 2007], le sudoku [Geem, 2007], la planification de tournées [Geem et al., 2005a], le partitionnement de pages web [Forsati et al., 2008], la conception de structures [Lee et al., 2004], la conception de réseaux hydrauliques [Geem, 2009], les tournées de véhicules [Geem et al., 2005], l'emploi du temps [Al-Betar et al., 2010], la segmentation d'image [Alia et al., 2009], etc.

2.5.2 Algorithmes à estimation de distribution

Plusieurs méthodes basées sur des modèles probabilistes ont été proposées [Larrañaga et Lozano, 2001]. Ces méthodes sont connues sous le nom d'algorithmes basés sur l'estimation de distribution.

Les algorithmes basés sur l'estimation de distribution reprennent le principe des algorithmes à population. Ils utilisent l'information globale contenue dans la population, mais utilisent des modèles probabilistes à la place d'opérateurs de mutation et de croisement pour construire de nouveaux individus.

Algorithme 10 : Pseudo-code de l'Algorithme à Estimation de Distribution

Initialisation de la population
Tant que le critère d'arrêt n'est pas atteint, **faire**
    Sélection des individus prometteurs
    Estimer la distribution de ces individus (construction d'un modèle probabiliste)
    Générer de nouvelles solutions à partir du modèle probabiliste
Fin Tant que

Le principe de fonctionnement de l'algorithme est le suivant [Yu, 2002]: On initialise tout d'abord aléatoirement une population D_0. Ensuite, à l'itération i, on utilise un opérateur de sélection pour la sélection des individus prometteurs et constituer la population D_{i-1}^{se}. L'information apportée par cette population est alors utilisée pour estimer la distribution de probabilité pl(x) de ces individus. Enfin, la nouvelle population D_i est créée aléatoirement à partir de la distribution de probabilité pl(x). Le parallèle avec les algorithmes évolutionnaires est ici respecté, le couple croisement-mutation étant remplacé par l'estimation de la distribution. La phase de diversification est ainsi assurée par l'utilisation d'une distribution de probabilité explicite, alors que la phase d'intensification n'est assurée que par la présence d'un opérateur de sélection.

On notera qu'une difficulté de cette méthode est le choix et l'estimation de la distribution de probabilité. En effet, il faut choisir au préalable quel modèle de distribution va être utilisé par l'algorithme et, à chaque itération, estimer les paramètres de cette distribution. Un pseudocode de l'algorithme à estimation de distribution est résumé par l'algorithme 10.

2.5.3 L'optimisation par colonies de fourmis

Les algorithmes à base de colonies de fourmis ont été introduits par Dorigo et al. [Dorigo et al., 1991] et dans la thèse [Dorigo, 1992]. L'optimisation par colonies de fourmis étudie le comportement de fourmis à la recherche de nourriture au départ de leur nid. Les algorithmes de colonies de fourmis sont nés d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis

communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol. Ce type de communication indirecte est appelé stigmergie.

Cette métaheuristique imite le comportement de fourmis cherchant de la nourriture. A chaque fois qu'une fourmi se déplace, elle laisse sur la trace de son passage une odeur (la phéromone). Comme la fourmi est rarement une exploratrice isolée, avec plusieurs de ses congénèress, elle explore une région en quête de nourriture. Face à un obstacle, le groupe des fourmis explore les deux côtés de l'obstacle et se retrouvent, puis elles reviennent au nid avec de la nourriture. Les autres fourmis qui veulent obtenir de la nourriture elles aussi vont emprunter le même chemin. Si celui-ci se sépare face à l'obstacle, les fourmis vont alors emprunter préférablement le chemin sur lequel la phéromone sera la plus forte. Mais la phéromone étant une odeur elle s'évapore. Si peu de fourmis empruntent une trace, il est possible que ce chemin ne soit plus valable au bout d'un moment, il en est de même si des fourmis exploratrices empruntent un chemin plus long (pour le contournement de l'obstacle par exemple). Par contre, si le chemin est fortement emprunté, chaque nouvelle fourmi qui passe redépose un peu de phéromone et renforce ainsi la trace, donnant alors à ce chemin une plus grande probabilité d'être emprunté. Il a été démontré expérimentalement (figure 2.8) que ce comportement permet l'émergence des chemins les plus courts entre le nid et la nourriture, à condition que les pistes de phéromones soient utilisées par une colonie entière de fourmis [Dréo et al., 2003]. Le premier algorithme s'inspirant de cette analogie a été proposé par Colorni, Dorigo et Maniezzo [Colorni et al., 1992].

L'une des applications principales de la méthode initiale était de résoudre le problème du voyageur de commerce. Chaque fourmi constitue un tour en empruntant des arêtes du graphe, puis au bout d'un moment les arêtes qui conduisent à une bonne solution deviendront les plus empruntées jusqu'à obtention d'une très bonne solution.

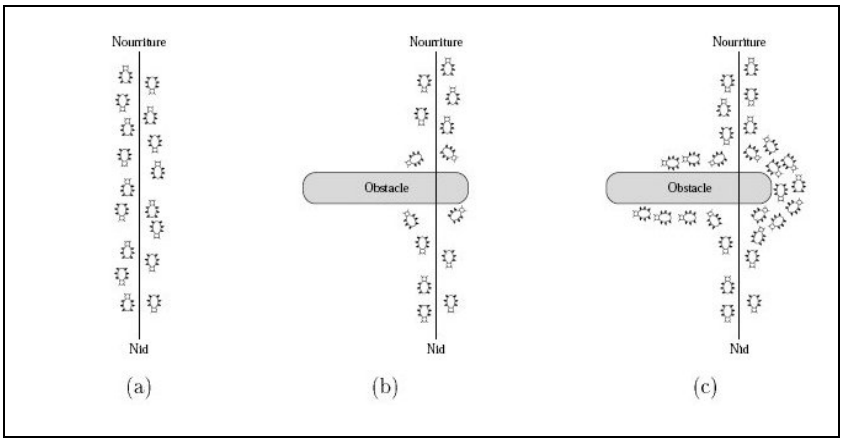


Figure 2.8 - Détermination du plus court chemin par une colonie de fourmis. (a) Situation initiale : les fourmis suivent une piste de phéromones entre la fourmilière et la nourriture. (b) Introduction d'un obstacle : les fourmis choisissent entre prendre à gauche et à droite avec équiprobabilité. (c) La phéromone s'évapore sur le chemin le plus long, les fourmis choisissent le chemin le plus court.

Même si le principe est simple, on peut voir apparaître des difficultés comme, la quantité de phéromone à déposer, le coefficient d'évaporation, la définition du choix biaisé, etc. La métaheuristique telle qu'elle est décrite schématiquement ci- dessous est détaillée dans [Dorigo et al., 1999]et [Dorigo et Di Caro, 1999]

Algorithme 11 :Pseudo-code de l'Algorithme de Colonies de Fourmis

Initialisation des pistes de phéromone
Tant que le critère d'arrêt n'est pas atteint **faire**
Pour chaque fourmi **Faire**
    Construction de solution (pistes-phéromone);
Mise à jour des pistes de phéromone :
Evaporation ;
Renforcement ;
Fin pour
Fin Tant que

L'algorithme comprend deux étapes itératives à savoir : la construction de solution et la mise à jour de phéromone. La construction de solution se fait selon une loi de transition d'états probabiliste. Elle permet de rajouter des composants de solution jusqu'à l'obtention d'une solution complète. Ce processus se base essentiellement sur les pistes de phéromone. Après chaque génération de solution, les pistes de phéromones sont mises à jour en utilisant la règle d'évaporation de phéromone (cette mise à jour a pour fonction de diminuer la quantité de phéromone de manière automatique, chaque valeur de phéromone est réduite par une même proportion : $\tau_{ij}=(1-\rho)\ \tau_{ij}\ \forall i,j \in [1,n]$ et $\rho \in]0,1]$), ou la règle de renforcement de phéromone (la quantité de phéromone est augmentée selon la qualité de la solution générée). Les facteurs d'intensification et de diversification sont plus difficiles à cerner. Néanmoins, le choix d'un nouvel attribut se fait aléatoirement (même s'il est biaisé), c'est donc un facteur de diversification. L'intensification se retrouve dans les paramètres de dépôt et d'évaporation qui concentrent plus ou moins les teneurs en phéromone des attributs de la solution.

Depuis, la démarche initiée par cette analogie a été étendue à la résolution d'autres problèmes d'optimisation, discrets et continus [Dorigo et al., 2005 ; Siarry et al., 2006]. Les algorithmes de colonies de fourmis présentent plusieurs caractéristiques intéressantes, telles que le parallélisme intrinsèque élevé, la robustesse (une colonie peut maintenir une recherche efficace même si certains de ses individus sont défaillants) ou encore la décentralisation (les fourmis n'obéissent pas à une autorité centralisée). De ce fait, ces algorithmes semblent être idéalement adaptés aux problèmes dynamiques, pour lesquels seule une information locale est disponible [Tfaili, 2007].

2.5.4 Les algorithmes particulaires

L'optimisation par essaim particulaire est une méthode née en 1995 aux USA sous le nom de Particle Swarm Optimization (PSO) par Russel Eberhat et James Kennedy [Kennedy et al., 1995].Cette méthode est inspirée du comportement social des animaux évoluant en essaim. L'exemple le plus souvent utilisé est le comportement des bancs de poissons [Reynolds, 1987]. En effet, on peut observer chez ces animaux des dynamiques de déplacementrelativement complexes, alors qu'individuellement chaque individu a une intelligence limitéeet une connaissance seulement locale de sa situation dans l'essaim. Un individu de l'essaim n'a pour connaissance que la position et la vitesse de ses plus proches voisins. Chaque individu utilise donc, non seulement, sa propre mémoire, mais aussi l'information locale sur ses plus proches voisins pour décider de son propre déplacement. Kennedy et Eberhart se sont inspirés de ces comportements socio-psychologiques pour créer PSO.

Un essaim de particules, qui sont des solutions potentielles au problème d'optimisation,survole l'espace de recherche de dimension D, en quête de l'optimum global. Le déplacement d'une particule est influencé par trois composantes (Figure 2.9) :

- Une composante physique : la particule tend à suivre sa direction courante de déplacement ;
- Une composante cognitive : la particule tend à se diriger vers le meilleur site par lequel elleest déjà passée ;
- Une composante sociale : la particule tend à se fier à l'expérience de ses congénères et, ainsi, à se diriger vers le meilleur site déjà atteint par ses voisins.

La qualité d'un site de l'espace de recherche est déterminée par la valeur de la fonction objectif en ce point.

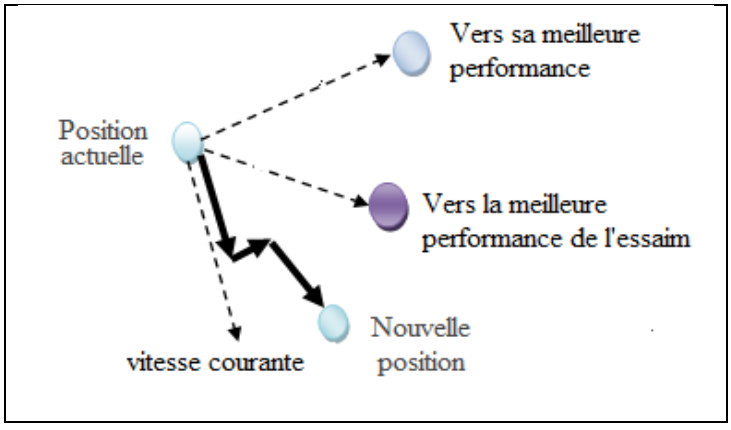


Figure 2.9- Mouvement d'une particule et la mise à jour du vecteur vitesse.

Il s'agit alors de modéliser des interactions sociales entre des *agents* devant atteindre un objectif donné dans un espace de recherche commun, chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information. La règle de base reside dans le fait qu'il n'y ait aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations, seulement des connaissances locales. Ainsi, une particule a sa propre *position* et *vitesse* (direction et distance à parcourir). L'optimisation est obtenue par la coopération des particules, le succès de quelques particules influence le comportement de leur cammarades.

Chaque particule i ajuste constamment sa position x_i vers l'optimum global et ce, en tenant compte de sa meilleure position jusque là visitée :$p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ et de la meilleure position visitée par l'essaim : $g = (g_1, g_2, \ldots g_D)$ [Talbi, 09]. Au temps t, le vecteur vitesse est calculé à partir de l'équation

$$v_i(t) \leftarrow w * v_i(t) + \rho_1 C1\big(p_i - x_i(t)\big) + \rho_2 C2(g - x_i(t)) \qquad (1)$$

La vitesse d'une particule représente la direction ainsi que la distance que la particule devra emprunter et parcourir. Les paramètres ρ_1 et ρ_2 sont deux nombres aléatoires uniformément distribués dans l'intervalle [0, 1]. *C1* et *C2* représentent le taux d'attraction que la particule possède soit envers la meilleure solution visitée par elle-même ou celle explorée par l'ensemble des autres particules. Le paramètre w quant à lui a pour objectif de contrôler l'impact de l'ancienne vitesse sur la nouvelle, pour une grande valeur de w la particule se déplacera en grande vitesse ce qui en résulte une diversification dans la recherche tandis qu'une petite valeur de w encourage l'exploitation locale ce qui se traduit par une intensification dans la région locale [Dreo et al., 2006]. La position au temps t de la particule i est alors définie par :

$$x_i(t) \leftarrow x_i(t) + v_i(t) \qquad (2)$$

Algorithme 12 : Pseudo-code de l'algorithme d'Optimisation par Essaim Particulaire

Initialisation aléatoire des positions et des vitesses de chaque particule
Pour chaque particule i, $p_i = x_i$
Tant que le critère d'arrêt n'est pas atteint **faire**
  Pour i = 1 à N **faire**
    Déplacement de la particule
    Évaluation des positions
    Si f (x_i) < f (p_i)
      $p_i = x_i$
    Fin Si
    Si f (p_i) < f (g)
      $g = p_i$
  Fin Pour
Fin Tant que

PSO est un algorithme à population de solutions. Il commence par une initialisation aléatoire de l'essaim dans l'espace de recherche. A chaque itération de l'algorithme, chaque particule est déplacée suivant (1) et (2). Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées. Les vecteurs P_i et G sont alors mis à jour. Cette procédure est résumée par l'Algorithme 7 (N est le nombre de particules de l'essaim).

Il est possible que le déplacement d'une particule la conduise à sortir de l'espace de recherche. Supposons, par simplicité, que l'espace de recherche soit [X*min, Xmax*]. Ce mécanisme stipule que si une coordonnée X_d, calculée selon les équations de mouvement, sort de cet intervalle, on lui attribue la valeur du point frontière le plus proche. Ceci est exprimé par la formule suivante :

$$x_i \leftarrow \mathrm{Min}(\mathrm{Max}(x_i + v_i), x_{max})$$

De plus, on complète souvent le mécanisme de confinement par une modification de la vitesse, soit en remplaçant la composante qui pose problème par son opposée, souvent pondérée par un coefficient inférieur à 1, soit, tout simplement en l'annulant.

Ce paragraphe a présenté succinctement la métaheuristique « Particle Swarm Optimization «. Pour plus de détails sur la méthode, une revue complète est disponible dans [Banks et al., 2007].

2.6 Conclusion

De nombreux problèmes réels peuvent s'exprimer comme des problèmes d'optimisation combinatoire. Or un grand nombre d'entre eux sont NP-difficiles et ne pourront donc pas être résolus de manière exacte dans un temps raisonnable. Lorsqu'on s'attaque à des problèmes réels, il faut se rendre à un compromis entre la qualité des solutions obtenues et le temps de calcul utilisé.

Les métaheuristiques constituent une classe de méthodes approchées adaptables à un grand nombre de problèmes d'optimisation combinatoire. Leur utilisation dans de nombreuses applications montre leur efficacité à résoudre les problèmes importants et complexes. Ce succès tient pour une grande part à leur capacité à fournir des solutions dont la qualité est au-delà de ce qu'il aurait été possible de réaliser avec une simple heuristique.

Une grande variété de métaheuristiques a été développée. Elles reposent sur un ensemble commun de concepts peu nombreux : la mémoire, qui sauvegarde l'information recueillie par l'algorithme, l'intensification qui tente d'améliorer la pertinence des informations disponibles, au

moyen de recherches locales, et la diversification, qui vise à accroître la quantité de ces informations, en explorant de nouvelles régions dans l'espace de recherche.

Dans le chapitre suivant, nous considérons le problème de détection d'intrusion par l'analyse des traces d'audit de sécurité. C'est un problème NP-Difficile [Mé, 1994]. Il y a alors nécessité d'utiliser les métaheuristiques pour sa résolution. Dans ce cadre, nous considérons les métaheuristiques « Biogeography Based Optimisation » et « Harmony Search » qui sont des algorithmes évolutionnaires, performants pour la résolution de problèmes sous contraintes. Elles représentent les moteurs d'analyse dans des systèmes de détection d'intrusion. Chacune d'elles utilise des paramètres, comme pour la plupart des métaheuristiques, nécessitant alors leur réglage (détermination des valeurs des paramètres pour lesquels les métaheuristiques donnent les meilleurs résultats possibles).

Chapitre 3

Problème d'analyse des traces d'audit de sécurité par les métaheuristiques

3.1 Introduction

Aujourd'hui, les systèmes d'informations et les réseaux informatiques occupent une place centrale dans la société moderne. Ils sont devenus des outils indispensables pour le bon fonctionnement et l'évolution de la plupart des entreprises. L'interconnexion de plus en plus importante de ces divers systèmes et réseaux a rendus ces derniers accessibles par une population diversifiée d'utilisateurs qui ne cesse d'augmenter. Ces utilisateurs, connus ou non, ne sont pas forcément pleins de bonnes intentions vis-à-vis de ces réseaux. Ils peuvent essayer d'accéder à des informations sensibles pour les lire, les modifier ou les détruire ou encore tout simplement pour porter atteinte au bon fonctionnement du système. Dès lors que ces réseaux sont apparus comme des cibles d'attaques potentielles, les sécuriser est devenu un enjeu incontournable.

Les systèmes de sécurité ont donc pour objectif la protection des données et informations. Ils sont chargés de protéger les droits de propriété, l'intégrité et la disponibilité des données. Beaucoup d'efforts ont été faits pour atteindre cet objectif: les politiques de sécurité, pare-feu, systèmes de détection d'intrusion (IDS), les logiciels anti-virus et les normes de configuration des services dans les systèmes d'exploitation et des réseaux [Bace, 2000].

La sécurité des réseaux et systèmes peut se faire, soit de manière préventive, soit de manière réactive. Dans l'approche préventive, il s'agit de protéger les données et ressources du système ou réseau contre tout accès non autorisé ou abusif. Cependant, prévenir de toutes les violations de sécurité, apparaît quelque peu irréel, et il est pratiquement impossible d'avoir un réseau complètement sûr et de le protéger contre toutes les attaques possibles. L'approche réactive, elle, consiste à essayer de détecter ces attaques au plus tôt afin de réagir rapidement et d'éviter ainsi que de sérieux dommages soient causés. Cette seconde approche, qualifiée de détection d'intrusions par l'analyse des traces d'audit de sécurité, fera l'objet de ce chapitre.

L'étude de méthodes efficaces pour détecter les intrusions dans les fichiers de traces d'audit est un élément important du vaste effort visant à améliorer les systèmes de détection d'intrusion. La majorité de ces méthodes sont basées sur des techniques d'intelligence artificielle [Wu et Banzhaf, 2010]. Différentes méthodes ont été proposées incluant les réseaux de neurones, les systèmes immunitaires et les algorithmes génétiques.

Deux nouveaux outils de détection d'intrusions basés sur l'analyse de fichiers de trace d'audit de sécurité sont proposés dans le présent chapitre. Ils suivent les lignes directrices d'un IDS proposé par Ludovic Mé [Mé, 1998], et utilisent des métaheuristiques relativement récentes : Biogeography Based Optimization (BBO) [Simon, 2008] et Harmony Search (HS) comme moteur d'analyse. BBO et HS sont deux algorithmes évolutionnaires à population de solutions bien conçus pour les problèmes d'optimisation sous contraintes. Une présentation de BBO et de HS a été faite dans le chapitre 2. Des notions portant sur les systèmes de détection d'intrusions ainsi que sur l'audit de sécurité sont décrites dans les sections 3.2 et 3.3.

3.2 Les systèmes de détection d'intrusions

3.2.1 Les intrusions

Une intrusion est toute violation de la sécurité logique d'un système informatique, c'est-à-dire une violation de l'une des trois propriétés que sont : la confidentialité, l'intégrité ou de disponibilité du système [Anderson, 1980]. Ces propriétés constituent les trois principaux axes de la sécurité informatique.

La confidentialité se réfère au fait que l'information est seulement connue par les utilisateurs autorisés. L'atteinte à la confidentialité se réaliserait par :

- Le vol de données : lecture, copie ou prise (copie suivie d'une destruction) ;
- Le furetage : parcours des répertoires à la recherche d'informations dont on ne connait pas forcément l'existence ;
- La fuite d'informations par canal caché ;
- L'inférence illégitime : corrélation illégitime entre des données pour lesquelles on a des droits d'accès ;
- L'introduction de programmes malveillants : Cheval de Troie.

L'intégrité signifie que les informations sont protégées contre toute altération. L'atteinte à l'intégrité se réaliserait par la fraude : modification illégitime de fichiers de données.

Quant à la disponibilité, elle signifie que le système fonctionne comme il a été conçu, que les utilisateurs y ont accès quand ils en ont besoin, où ils en ont besoin et dans la forme voulue. L'atteinte à la disponibilité de service se réaliserait par :

- La destruction illégitime ou abusive de fichiers : destruction de données ou de programmes ;
- L'occupation illégitime ou abusive de ressources : avec ou sans bénéfices d'usage ;
- La réduction illicite ou abusive des droits des usagers ;
- L'introduction de programmes malveillants : virus, vers, etc.

Plus généralement, une intrusion peut être définie comme un abus ou une activité anormale. L'abus se réfère aux tentatives d'exploiter les points faibles dans l'ordinateur ou à l'abus de privilèges d'un système existant. Une activité anormale représente des déviations significatives d'une opération normale du système ou d'utilisation par les utilisateurs du système [Tadjen, 2004].

3.2.2 La détection d'intrusion

La détection d'intrusions est le processus de suivi des événements survenus dans un système ou un réseau, et l'analyse de ces événements pour trouver des signes d'intrusions. Il s'agit d'identifier les individus utilisant le système sans avoir l'autorisation (hackers) et les individus ayant un accès légitime au système mais qui abusent de leurs privilèges (insiderthreat) [Mukherjee et al., 1994], [Scarfone et Mell, 2007]. Elle a été introduite en 1980 par J.P Anderson [Anderson, 1980], mais n'a pas eu beaucoup de succès à ses origines. Il a fallu attendre la publication d'un modèle de détection d'intrusion par Denning en 1987 [Denning, 1987] pour marquer réellement le départ du domaine. La recherche dans le domaine s'est ensuite développée et le nombre de prototypes s'est énormément accru. Plusieurs domaines tels que le data mining, les agents, réseaux de neurones, l'immunologie ou encore les algorithmes génétique, qui ont montré leurs performances dans la détection d'attaques et intrusions potentielles, sont impliqués dans la réalisation de système de détection d'intrusion [Biondi, 2001].

3.2.3 Les systèmes de détection d'intrusion

Les systèmes de détection d'intrusion (IDS) constituent l'une des premières approches du problème de la sécurité informatique. Ce sont des logiciels et des matériels qui automatisent le processus d'observation et d'analyse des événements. Ils surveillent dynamiquement les événements prenant place dans un système, et décident si ces événements sont symptomatiques d'une attaque ou constituent une utilisation légitime du système [Dass et Lids, 2003]. Pour être efficaces, ils doivent s'exécuter constamment sur le système, en travaillant en arrière - plan, et ne notifiant l'administrateur de sécurité que lorsqu'ils détectent quelque chose qu'ils considèrent comme suspicieux ou illégal [Price, 1998]. Ils doivent s'adapter aux changements de comportements et aux grandes quantités de données, être configurables, ne pas utiliser trop de ressources mémoires de la machine et après les pannes de système, être réutilisables sans nouvel apprentissage [Balasubramaniyan et al.,, 1998].

Tous les IDS prennent en compte sous une forme ou une autre l'hypothèse que l'exploitation des vulnérabilités d'un système est basée sur l'utilisation anormale du système [Denning, 1987].

Plusieurs schémas, décrivant les composants d'un système de détection d'intrusion ont été proposés. Selon Intrusion Detection Working Group (IDWG), l'architecture d'un système de détection d'intrusion comporte 3 modules [Wood et Erlinger, 2007]: les capteurs, l'analyseur et le manager, lesquels interagissent. Les capteurs ont pour rôle d'extraire et de transformer certaines informations fournies par le système (activité système) et de les transmettre sous forme d'événements à un analyseur. Ce dernier utilise ces événements afin de déceler une possible intrusion et génère en conséquence des alertes. Un ou plusieurs capteurs couplés à un analyseur forment une sonde. Le manager reçoit des alertes venant d'une sonde. Il est alors chargé de les traiter et de notifier toute activité suspecte sur le système, à l'opérateur de sécurité. La Figure 3.1 illustre de manière simplifiée ces différents modules et leurs interactions [Michel, 2003].

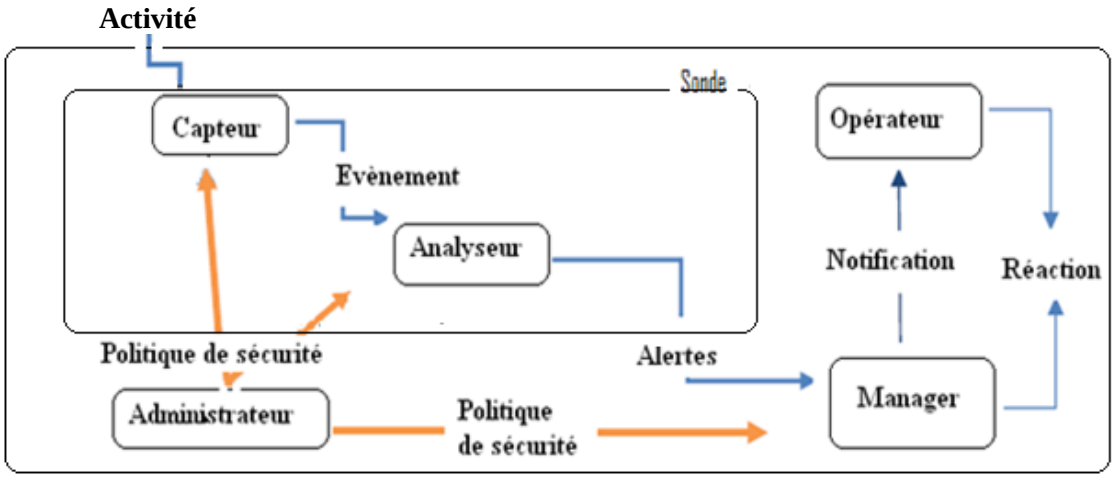


Figure 3.1 - Architecture de l'IDWG d'un système de détection d'intrusions.

Chaque IDS dispose de propriétés qui lui sont propres. Connaître ces différentes particularités permettra de choisir l'IDS adéquat à un système bien spécifique et d'en tirer le meilleur. Plusieurs classifications des IDS ont été réalisées dont la classification suivant l'approche de détection.

Tous les IDS prennent en compte sous une forme ou une autre l'hypothèse que l'exploitation des vulnérabilités d'un système est basée sur l'utilisation anormale du système [Denning, 1987]. Deux approches principales sont utilisées dans la conception des IDS : l'approche comportementale et l'approche par scénarios. Lorsque l'IDS utilise des informations sur le comportement normal du système qu'il surveille (approche comportementale), on dit qu'il est basé comportement. Lorsque l'IDS utilise des informations sur les attaques (approche par scénarios) on dit qu'il est basé sur la connaissance. Toutefois, les deux approches peuvent être complémentaires [Tombini, 2006].

L'approche comportementale a été la première approche proposée. Elle fut introduite par Anderson [Anderson, 1980], et développée par Denning [Denning, 1987]. Les comportements normaux des utilisateurs du réseau sont connus et il est donc possible de construire des profils

représentant ces comportements grâce à plusieurs caractéristiques comme les activités sur le réseau, etc [Dewan et al., 2009]. Une fois ces profils définis, les intrusions sont identifiées comme des déviations par rapport aux comportements normaux. Cette approche présente des avantages tels que la détection d'attaques inconnues, mais également des inconvénients. En effet, la méthode est sujette à plusieurs faux positifs (évènements normaux détectés comme attaques) si le modèle comportemental n'est pas complet. De même que si le modèle est construit à l'aide d'un algorithme d'apprentissage, et que durant la phase d'apprentissage un comportement malveillant se produit, ce dernier sera automatiquement inclus dans le modèle et ne déclenchera pas d'alerte durant la phase de recherche d'attaque (faux négatif). Les techniques développées [Wu et Banzhaf, 2010] incluent les systèmes experts, les modèles statistiques, et des systèmes immunitaires artificiels [Yang et al., 2011], [Banković et al., 2009]. D'autres techniques, comme l'estimation des paramètres bayésiens [Cha et SAD, 2004], le clustering [Xu et Zhang, 2005], [Sh et Ws, 2003] ainsi que les algorithmes génétiques [Ozyer et al.,2007], [Abadeh et al., 2007] sont également utilisées. Dans les « systèmes experts » par exemple, une base de règles, régulièrement rafraichie, décrit statistiquement le profil de l'utilisateur au vu de ses précédentes activités. Pendant la phase de détection, le système applique les règles au flux d'événements, et vérifie si ce flux d'événements respecte ou non les règles apprises [Majorczyk, 2008].

Dans la seconde approche, dite par scénarios (Misuse Detection), la détection d'abus est basée sur l'identification directe des attaques connues. Les modèles (signatures) d'attaques peuvent être créés par les experts du domaine [Dokas et al., 2002]. On recherche alors les occurrences de ces signatures dans un flux d'événements; si une concordance est trouvée, une alerte est lancée. De manière générale, une signature désigne un ensemble de caractéristique permettant d'identifier une activité intrusive : il peut s'agir d'une chaine alphanumérique, d'une taille de paquets inhabituelle, d'une trame formatée de manière suspecte, etc. [Evangelista, 2004]. Un détecteur d'intrusions par scénario comprend ainsi un ensemble de sondes produisant un flux d'évènements, une base de signatures d'attaques ainsi que d'un algorithme de recherche de motifs qui compare le flux d'événements aux signatures contenues dans la base. Les IDS traitent les enregistrements reçus du système d'exploitation pour une période de temps spécifique afin d'avoir un ensemble complet de l'activité des utilisateurs. L'IDS effectue alors une analyse de l'activité en cours, en utilisant un système de base de règles, les statistiques, ou une heuristique correspondante pour déterminer la présence éventuelle d'une intrusion [Srivastava et al., 2006].

Cette approche par scenario est simple et est basée sur des algorithmes performants (en temps de calcul ainsi qu'en consommation de mémoire). Cependant, elle présente également des inconvénients. En effet, seules les attaques connues sont détectées. De plus, la mise à jour de la base des signatures doit être permanente, car sinon on risque d'avoir plusieurs faux négatifs (attaques non détectées). Il est également difficile de distinguer entre les actions légales et les actions illégales à

cause de la complexité du système surveillé, ce qui risque de produire de fausses alertes. Dans cette approche, la reconnaissance de signatures utilise des algorithmes de type pattern matching, pour reconnaitre une signature dans une trace correspondant à une suite d'évènements. Le fichier d'audit peut être vu comme une chaine de caractères principale et les scénarios d'attaques comme des sous chaînes qu'il s'agit de localiser dans cette chaine principale. Cette approche pose un problème lorsque plusieurs scenarii donnent lieu à une même signature. Pour pallier ce problème, certaines approches utilisent des algorithmes de reconnaissance basés sur des algorithmes génétiques [Me, 98], des réseaux bayésiens [DuMouchel, 1999].

Certains IDS utilisent les statistiques, tandis que d'autres utilisent une approche d'apprentissage, comme des réseaux de neurones [Bace, 2000], le système immunitaire [Forrest et al., 1994], les algorithmes génétiques [Mé, 1998], [Diaz-Gomez et Hougen, 2006]], et [Ahmim et al., 2011] , la programmation génétique [Crosbie et Spafford, 1995].

3.3 Une approche de l'audit de sécurité

3.3.1 Introduction

L'audit de sécurité est un mécanisme qui permet d'enregistrer chronologiquement dans un fichier particulier appelé journal d'audit, toutes ou partie des actions (activités système), effectuées dans un système informatique donné, qu'il convient de protéger. Le journal d'audit doit permettre à partir de l'étude des séquences d'évènements, appelées traces d'audit, que l'on y trouve, la reconstruction des opérations effectuées, ainsi que les ressources du système qui ont été affectées, l'identification de l'horodatage des enregistrements, et en cas d'échec, pourquoi l'opération a-t-elle échouée ? [Mé, 1994]. Ces traces d'audit seront utilisées à posteriori afin de démasquer d'éventuelles intrusions et leurs auteurs [Tadjen, 2004]. Garder une trace de toutes les actions entreprises sur le système, permet non seulement de révéler toutes violations des règles de la stratégie de sécurité dressée par les administrateurs, mais aussi d'identifier les responsables des différentes actions litigieuses ou encore d'identifier d'éventuels dégâts.

L'analyse du fichier d'audit nécessite au préalable de mener deux actions : la spécification des utilisateurs ainsi que des activités à auditer, et ensuite, assurer la collecte de ces derniers dans le fichier. Parmi les informations auditables, figurent des informations sur l'accès au système, des informations sur l'usage fait du système et des fichiers, des informations sur les applications, des informations sur les violations éventuelles de la sécurité, des informations sur les performances du système (statistiques), etc [Brown, 1979]. Pour assurer la génération et la collecte de certains types d'évènements, les systèmes d'exploitation ainsi que les applications sont munis de sous systèmes d'audit, qui permettront à l'administrateur système d'analyser le fichier d'audit généré. Nous ne détaillons pas ici les différents mécanismes de collecte des évènements. Les systèmes sont différents,

le lecteur désirant avoir plus d'informations sur le mécanisme de collecte, peut se reporter au manuel d'utilisation du système qu'il va auditer. Dans le travail que nous présentons, nous utiliserons des traces d'audit déjà prêtes ou générées aléatoirement, car il s'agit pour nous de prouver les performances d'une méthode d'analyse de traces d'audit. Les résultats obtenus par cette analyse doivent permettre de trouver une solution aux dégâts occasionnés par les intrus : en tentant par exemple de les déconnecter ou de les confiner dans des répertoires particuliers, limitant ainsi les dommages possibles et permettant d'étudier plus précisément leurs comportements (cela n'étant pas toujours possible). Il faudrait pouvoir revenir à un état sain (d'avant l'intrusion). De ce fait, cette analyse nous permettra de cerner les points faibles et les failles de sécurité existants dans le système. A noter que l'écart entre deux analyses successives des traces d'audit doit être aussi petit que possible, et cela afin de détecter un maximum de transgressions.

2.3.2 Le modèle

L'audit de sécurité, à l'image du diagnostic médical, a pour objectif de déterminer l'ensemble des conditions qui peuvent expliquer la présence de symptômes observés. Pour cela, un expert humain utilise des connaissances spécifiques de type cause-à-effet (les symptômes sont les évènements enregistrés dans le fichier d'audit et les connaissances de l'expert sont les scénarios d'attaque). L'expert utilise ses connaissances pour élaborer des hypothèses qu'il confronte à la réalité observée. S'il reste des symptômes observés que l'hypothèse faite n'explique pas, ou si celles-ci engendrent davantage de symptômes que ceux observés dans la réalité, une nouvelle hypothèse plus pertinente doit être essayée. L'ordre d'apparition des symptômes n'est pas pris en compte. Dans notre cas, cela signifie que les scénarios d'attaque ne sont plus considérés comme des séquences d'événements, mais comme des ensembles de couples (E_i, n_i) où E_i représente un type d'événements et n_i le nombre d'occurrences de ce type d'événements dans le scénario. Ainsi, on peut définir la matrice AE dite matrice attaques-évènements de dimension n_e x n_a telle que AE_{ij} est le nombre d'évènements de type i, générés par l'attaque j, $AE_{i\,j} \geq 0$. L'ordre temporel des évènements n'est pas pris en considération. Il peut en résulter une altération de la pertinence de la détection. Cependant, on peut y voir un avantage car, dans la pratique, l'ordonnancement total des événements est bien souvent difficile, voire impossible. C'est particulièrement le cas lorsque l'unité de temps minimale offerte par le système d'audit est trop importante. Ainsi, si cette unité est la seconde, les nombreux événements qui surviennent en une seconde ne peuvent être ordonnés. Si l'audit s'étend à tout un réseau, le problème s'accentue puisqu'il faut alors construire un temps global, dont on sait que la précision sera limitée.

De plus, à chaque attaque, le système infecté encourt un certain risque qui est relatif à l'attaque. Ce risque sera représenté par un vecteur ligne d'entiers noté R de taille n_a. Un élément R_i de R représentera le poids de l'attaque i tel que $R_i > 0$. Pour des raisons de simplicité et afin de ne pas privilégier une attaque par rapport à une autre, les différents éléments de ce vecteur auront une valeur égale à 1.

Formellement, le problème d'analyse des traces d'audit de sécurité peut se présenter de la façon suivante [Mé, 1994], [Diaz-Gomez et Hougen, 2006] et [Ahmim et al., 2011] :

$$\begin{cases} \text{Max} \sum_{j}^{n_a} R_j . H_j \\ (AE . H)_j \leq O_j, \quad j = 1 \dots n_a \end{cases}$$

Le problème consiste alors à déterminer le vecteur H maximisant le produit R.H, tout en respectant les contraintes $(AE.H) \leq O_i$ tel que $i=1,..n_e$. Trouver le meilleur vecteur H revient à trouver le maximum du produit R.H parmi les différentes combinaisons de scénar2 d'attaques codés par les hypothèses.

Les hypothèses (vecteurs H) sont sujettes à une contrainte qui se résume comme suit : L'hypothèse qu'une attaque ait eu lieu stipule l'exécution des différents évènements la caractérisant. Ceci signifie que le nombre d'occurrence de ces derniers soit présent dans le fichier O. De ce fait, toute hypothèse qui coderait l'existence d'une attaque « i » (Hi =1) alors que cette dernière n'a pas eu lieu(attaque non existante dans le vecteur O) impliquerait que les évènements générés par l'attaque « i » ne soient pas présents dans le vecteur O et rend l'hypothèse non réaliste. Ainsi, si $((AE.H)_i > O_i)$ alors l'hypothèse n'est pas réaliste car le nombre d'occurrences de l'évènement i théorique (d'après l'hypothèse) est supérieur au nombre d'occurrence de l'évènement « i » effectif (dans O), et si ((AE.H)i > Oi) alors l'hypothèse est réaliste car le nombre d'occurrence de l'évènement « i » théorique (d'après l'hypothèse) est inférieur ou égal à celui enregistré dans O.

La Figure 3.2 suivante présente un schéma récapitulatif du problème d'analyse des traces d'audit, où :

- n_e : nombre de types d'évènements auditables
- n_a : nombre d'attaques connues
- AE : matrice attaques-évènements de dimension n_e x n_a telle que AE_{ij} est le nombre d'évènements de type i générés par l'attaque j($AE_{ij} \geq 0$)
- R : vecteur de dimension n_a, où R_i (Ri> 0) est le poids associé à l'attaque i (R_i est proportionnel au risque inhérent à l'attaque i).
- O : vecteur de dimension n_e où O_i est le nombre d'évènements de type i présents dans le fichier d'audit. (O est dit vecteur observé)
- H: vecteur de dimension n_a où $H_j = 1$ si l' attaque i is présente dans l'hypothèse et $H_j = 0$ sinon (H décrit un ensemble d'attaques particulier).

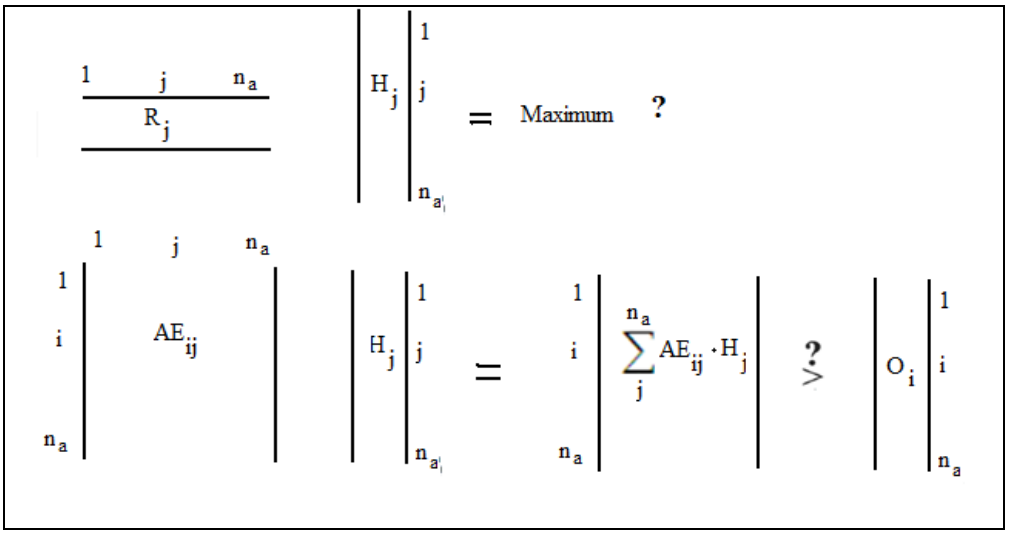


Figure 3.2 - Problème d'analyse des traces d'audit de sécurité

Pour expliquer les données contenues dans les traces d'audit (i.e. O) par la présence d'une ou plusieurs attaques, nous devons trouver le vecteur H qui maximise le produit R.H (c'est l'approche pessimiste: trouver H tel que le risque soit maximum), vérifiant les contraintes $(AE.H)_i \leq O_i$, $1 \leq i \leq n_a$

Trouver le vecteur H est un problème NP-Complet [Mé, 1994]. Aussi l'application des algorithmes classiques est impossible quand n_a est assez grand (de l'ordre de plusieurs centaines). Mé utilise les Algorithmes Génétiques [Mé, 1998].

Dans les différents travaux utilisant les algorithmes génétiques, les solutions sont codées dans des chaînes binaires ; la longueur d'une chaîne représente le nombre d'attaques, la valeur 1ou 0 dans le génome indique la présence ou non d'une attaque. La fonction objectif a été biaisée en faveur des individus capables de prédire un grand nombre de types d'intrusions (nombre de 1 dans les chromosomes), tout en évitant les alarmes d'attaques qui n'existent pas (inutile 1 dans les chromosomes). Diaz-Gomez et Hougen [Diaz-Gomez et Hougen, 2006] ont apporté des modifications dans la fonction objectif utilisée dans [Mé, 1994].

L'heuristique choisie pour résoudre ce problème NP-Complet est comme suit: une hypothèse est faite (e.g. parmi l'ensemble des attaques possibles, les attaques i, j et k sont présentes dans les traces). L'hypothèse est évaluée, et suivant sa valeur, une meilleure hypothèse est testée jusqu'à l'obtention d'une solution. Afin d'évaluer une hypothèse correspondant à un sous-ensemble particulier d'attaque présente, nous comptons la survenance d'événements de chaque type généré par toutes les attaques de l'hypothèse. Si chacun des chiffres est inférieur ou égal au nombre d'événements enregistrés, alors l'hypothèse est réaliste. Pour dériver une nouvelle hypothèse, une métaheuristique (BBO dans la première approche et HS dans la deuxième) est alors utilisée. La Figure 3.3 montre le modèle de détection d'intrusions utilisant BBO comme moteur d'analyse.

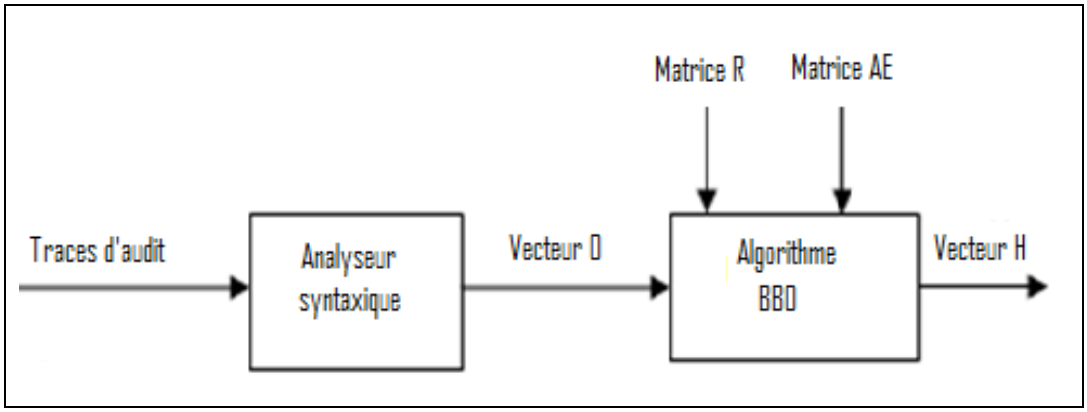


Figure 3.3 - Exemple de modèle de détection d'intrusions

3.3.3 Structures des données

Dans ses travaux [Mé, 1998], Mé considère un IDS host-based, et utilise les algorithmes génétiques (AG) comme moteur de détection d'intrusion. Un fichier audit est généré par une machine Sun pour analyse. La durée de la session d'audit a été choisie de 30 minutes. Quatre (4) types d'utilisateurs ont été définis: utilisateurs inexpérimentés, développeur débutant, professionnel, et utilisateur intense d'UNIX. La matrice AE (Attaques-Evènements) considérée (Figure 3.4) est de dimension 24x28 comprenant 24 scénarios d'attaques et 28 types d'évènements auditables propres à un système Unix.

La métaheuristique utilisée reçoit en données le vecteur observé O ainsi que la matrice AE des attaques prédéfinies. Dans la Figure 3.4, chaque ligne correspond à un type d'événement. Chaque colonne numérique - 0 à 23- correspond à une attaque connue. Le vecteur observé contient les numéros des événements- système effectués par un utilisateur lors d'une session. Comme montré dans le tableau TAB 1, la première ligne est le type d'événement et la deuxième rangée est le nombre d'événements de ce type. La première position de la ligne 2 indique un 1, ce qui signifie qu'il y avait un type d'événement : *User login fail*. La huitième position indique 40, ce qui signifie qu'il ya 40 événements de type *ls fail*. De même, dans TAB 2, les positions 4, 5, et 8 affichent un 1 indiquant la survenue possible des attaques 4, 5 et 8.

TAB 1. Exemple d'un Vecteur Observé

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	12	6	0	4	9	11	40	34	9	5	45	0	0	2	7	0	29	0	0	0	0	0	0	0	0	0	0

TAB 2. Exemple de vecteur solution.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Evènements	Attaques																								
User_Login fail	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
User_Login (23h à 6h)	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Short_Session	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
User_SU OK	.	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
User_SU fail	.	.	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
who, w, finger, f, ps…	.	3	.	.	.	.	.	.	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
more, pg, cat, vi, …	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	5	.	.	
ls OK	.	.	.	.	.	30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
ls fail	.	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
df, hostname, uname	.	.	.	.	.	.	.	.	.	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
arp, netstat, ping	.	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	
ypcat	.	.	.	.	.	.	.	.	.	.	.	3	.	.	.	.	.	.	.	.	.	.	.	.	
lpr	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	10	1	.	.	.	.	.	.	.	
rm, mv	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	
ln	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	
whoami, id	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4
rexec, rlogin, rsh, …	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	
Proc_Execute	.	3	.	.	.	35	5	.	8	3	2	3	.	.	10	3	.	300	.	2	.	5	.	4	
Proc_SetPri	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	100	.	.	.	.	.	
File_Open fail	.	.	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
File_Open fail cp	.	.	.	.	.	.	.	.	.	.	.	.	.	10	.	.	.	.	.	.	.	.	.	.	
File_Open .netrc	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	
File_Read lpr	.	.	.	.	.	.	.	.	.	.	.	.	.	.	10	.	.	.	.	.	.	.	.	.	
File_Read passwd, …	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	5	.	.	
File_Write passwd, … fail	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	
File_Write cp OK	.	.	.	.	.	.	.	.	.	.	.	.	30	.	.	.	.	.	.	.	.	.	.	.	
File_Unlink rm	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	50	.	.	.	.	.	.	
File_Mode	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	3	.	.	.	

Figure 3.4 - Matrice Attaques-Evénements (AE)

3.4 Analyse des traces d'audit de sécurité basée sur BBO

3.4.1 Adaptation de BBO

Cette approche a pour objectif de déterminer si les évènements générés par l'utilisateur correspondent à des attaques connues. Il s'agit alors de rechercher dans le fichier des traces d'audit de sécurité les occurrences d'attaques en utilisant une métaheuristique (BBO) car cette recherche constitue un problème NP-Complet. Le but de cette heuristique est l'obtention du vecteur Hypothèse H qui maximise le produit R.H sous les contraintes : $(AE.H)_i \leq O_i$, $1 \leq i \leq n_a$), où R est le vecteur poids qui reflète les priorités du manager de sécurité, AE est la matrice attaques- évènements qui corrèle les ensembles d'évènements avec les attaques connues,$1 \leq i \leq n_a$, et n_a est le nombre d'attaques connues.

BBO est un algorithme d'optimisation stochastique basé sur une vision relativement simpliste de la biogéographie. Elle travaille sur un ensemble d'individus (habitats ou îles) d'une population (archipel). Les îles sont caractérisées par un ensemble de SIV.

Pour utiliser BBO, nous devons d'abord trouver un bon encodage des solutions ainsi qu'une fonction objectif pour nous permettre leur évaluation. Une solution potentielle du problème à résoudre

est un vecteur H, exprimé sous forme d'une suite binaire (avec pour valeur 1 si l'attaque i est présente dans le fichier d'audit O et 0 sinon). Aussi, dans notre cas, l'encodage est direct : un habitat est composé de n_a SIV. La valeur de chaque SIV est 1 ou 0 suivant que l'attaque est présente ou non, et n_a est le nombre d'attaques. Chacun des habitats da l'archipel correspond à un vecteur H particulier (Figure 3.5).

De plus, comme spécifié ci-dessus, la recherché se fera parmi tous les sous ensembles d'attaques possibles pour déterminer celui qui présente le plus de risque possible pour le système. Ce qui se traduit par la maximisation du produit RxH. Comme BBO est un algorithme d'optimisation, qui détermine le maximum de la fonction objectif, nous pouvons rapidement conclure que dans notre cas cette fonction, notée F, doit être prise égale au produit R x H, soit

$$F = \sum_{i}^{n_a} R_i \, . H_i$$

où H est un habitat. Cependant, cette fonction ignore qu'il s'agit d'un problème sous contraintes, à savoir certains habitats parmi les 2^{na} sont non réalistes (solutions non réalisables), ce qui est le cas pour des évènements i tels que $(AE.H)_i > O_i$.

Les solutions ne satisfaisant pas les contraintes sont pénalisées en fixant leur valeur à 0, le taux de migration μ à 0, et le taux d'immigration λ à 1. Quant aux solutions vérifiant les contraintes, elles seront triées suivant la valeur de leur fonction objective par ordre décroissant. Un nombre d'espèces est associé à chacune des solutions. P (taille de la population) espèces sont associées à la meilleure solution, (P-1) espèces à la solution voisine, et ainsi de suite.

Une population initale de plusieurs habitats (vecteur binaire H) est générée aléatoirement. Elle évolue suivant un processus de migration (émigration et immigration) et un processus de mutation pour atteindre une solution optimale.

Le processus de migration dans le mécanisme de biogéographie a pour but de modifier une ou plusieurs SIV, choisies aléatoirement dans chaque solution considérée, permettant l'échange d'information entre les bonnes et mauvaises solutions.

La mutation maintient la diversité dans la population et explore l'espace de recherche, évitant à l'algorithme de converger trop rapidement vers un optimum local. La mutation vient après le processus de migration. Elle sera appliquée à la deuxième moitié de la population, avec une probabilité très faible (taux de mutation : pm=0.005), car une valeur forte pourra causer la perte d'information importante contenue dans les solutions. Au cours de ce processus, les caractéristiques (les SIV) d'un ou de plusieurs individus (habitats) de la population seront modifiées aléatoirement. En d'autres termes, une ou plusieurs caractéristiques (SIV) sont modifiées pour changer une solution en une autre de valeur différente.

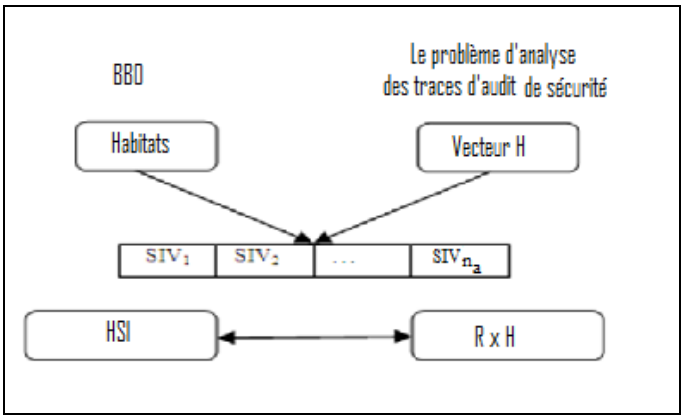


Figure 3.5- Représentation des solutions et de la fonction objectif

Cependant, l'individu modifié, ne sera pas nécessairement meilleur ou plus mauvais, mais il fournira plus d'opportunités pour créer de bonnes solutions. Au cours de l'évolution de la population, il se pourrait que les meilleures solutions soient modifiées, et donc perdues après les processus de migration et mutation. Pour éviter une telle situation, un opérateur d'élitisme est adopté. Il copie les p meilleures individus dans la nouvelle génération (la valeur de p est fixée par simulation).

Les processus de mutation et migration sont les mécanismes de BBO utilisés pour déduire une nouvelle hypothèse à partir de la précédente, en explorant l'espace de recherche, essayant d'améliorer la meilleure solution en fonction de la valeur de la valeur de la fonction objective (le risque encouru). Changer un SIV dans un vecteur H au moyen des processus de mutation et de migration provoque l'apparition ou la disparition d'une attaque. L'algorithme 13 montre les différentes étapes suivies dans l'analyse des traces d'audit de sécurité en utilisant l'algorithme BBO.

Algorithme 13: Pseudo-code du processus d'analyse des traces d'audit de sécurité avec BBO

Entrée: matrice AE et vecteur O
Initialiser les paramètres
Générer une population aléatoire d'îles (N sous-ensembles d'attaques)
Tester la faisabilité des solutions générées
Evaluer la fitness des solutions générées (Calcul du produit R*H)
Ordonner la population par ordre décroissant
Pour I = 1 à Nb_ Générations **Faire**
Sauvegarder les meilleures solutions de la génération courante
Calculer le nombre d'espèces
Calculer les taux d'émigration (α) et d'immigration (β)
Appliquer le processus de Migration
Appliquer le processus de Mutation
Tester la faisabilité des solutions générées
Calculer les valeurs de la fitness (produuit R*H)
Ordonner la population par ordre décroissant
Remplacer les mauvaises solutions de la génération présente par les individus sélectionnés de la génération précédente
Fin Pour
Sortie: Le meilleur vecteur H qui maximise le produit RxH et vérifie les contraintes.

Remarque : On notera que notre approche est sujette à des limitations telles que:

- Les attaques caractérisées par une absence d'évènements ne sont pas prises en considération.
- La réalisation multiple d'une attaque particulière n'est pas détéctée avec un codage binaire.
- Seules les attaques indépendantes sont considérées.
- La chronologie des évènements n'est pas prise en compte.

3.4.2 Mesures d'évaluation

Pour pouvoir évaluer la qualité de détection de la méthode proposée et analyser l'influence de chacun des paramètres sur cette qualité de détection, des mesures ont été utilisées. Il s'agit des ratios TNR, TPR, FPR, FNR [Weise et al., 2008] définis ci-dessous :

- TNR (True négative rate): TPR = TN / (TN+FP)
- TPR (True positive rate) : TPR = TP / (TP+FN)
- FPR (False positive rate) : FPR = FP / (TN+FP)
- FNR (False négative rate): FNR = FN / (TP+FN)
- Accuracy = (TN+TP) / (TN+TP+FN+FP)
- Précision = TP / (TP+FP)

où :

- TN est le nombre d'événements normaux détectés comme des événements normaux.
- TP est le nombre d'attaques détectées et réellement présentes dans le fichier d'audit
- FP est le nombre d'événements normaux détectés comme des attaques.
- FN est le nombre d'attaques prédites comme des événements normaux.

TN et TP correspondent à une détection d'intrusion correcte. Les évènements sont labélisés comme normaux ou comme attaques, respectivement, avec succès. FP se réfère aux événements normaux détectés comme attaques; FN correspond au nombre d'attaques incorrectement détectées comme normales. TPR définit alors le taux d'attaques détectées et effectivement présentes dans le fichier d'audit, FPR le taux d'attaques détectées mais absentes en réalité du fichier d'audit, ce taux correspond à l'évaluation des faux positifs. L'accuracy représente le taux de détection correcte des attaques, de même que precision définit le taux d'attaques présentes par rapport aux attaques détectées.

3.4.3 Tests et Résultats

Pour évaluer notre approche de détection d'intrusion avec BBO, nous utiliserons dans un premier temps le benchmark [Mé, 1998]. Nous étendrons ensuite notre étude à des instances plus larges en générant aléatoirement la matrice Attaques-Evènements.

Durant les tests, l'ensemble des attaques réellement présentes dans le fichier d'audit analysé doit être connu en avance. Ainsi, les évènements correspondant à une ou plusieurs attaques sont inclus dans le vecteur audit observé.

Les valeurs des mesures de performances: TPR, FPR, Précision, Accuracy et Specificité, sont calculées comme valeurs moyennes sur 10 exécutions. Cette évaluation est alors réalisée pour plusieurs valeurs de chaque paramètre de la métaheuristique utilisée. Ces paramètres sont : le nombre de générations (N), la taille de la population (P), le paramètre de mutation P_m, le paramètre d'élitisme (L), ainsi que le nombre d'attaques injectées (I_a).

Dans la suite, nous considérons le 5-uplet : (N =100, P =50, P_m = 0.005, L =2, i_a =2) comme le vecteur des valeurs par défaut des paramètres dont nous voulons mesurer l'influence sur la qualité de détection du système étudié.

3.4.3.1 Benchmark

Différents tests ont été réalisés. Les résultats sont reportés dans les tableaux TAB 3 à TAB 7.

Dans le tableau TAB 3, nous observons que pour différentes valeurs du nombre de générations N= (50, 100 et 200), 100% des attaques injectées sont détectées, et il n'y a pas de faux positif détecté : TPR = Accuracy = Specificité = 100% et FPR = 0.

Quand nous varions la taille de la population P= (50, 100 and 200), nous obtenons les mêmes bons résultats, comme indiqués dans le tableau TAB 4.

Ainsi, il n'ya pas lieu d'augmenter les valeurs de ces deux éléments ; Nous retenons les valeurs N=50 et P=50.

Dans le tableau TAB 5, nous observons l'influence du taux de mutation sur la qualité de la détection d'intrusion. Nous obtenons de bons resultants pour un taux de mutation de 0.005. Toutes les attaques sont détectées. TPR=100%. Les mêmes résultats sont obtenus en augmentant ou en diminuant la valeur de Pm. Cependant, les attaques ne sont pas détectées quand le taux de mutation n'est pas utilisé (nul). Nous retenons Pm = 0.005.

TAB 3. Influence du nombre de générations (N)

N	TPR	FPR	Precision	Accuracy	TNR
50	100%	0%	100%	100%	100%
100	100%	0%	100%	100%	100%
200	100%	0%	100%	100%	100%

TAB 4. Influence de la taille de la population (P)

P	TPR	FPR	Precision	Accuracy	TNR
50	100%	0%	100%	100%	100%
100	100%	0%	100%	100%	100%
200	100%	0%	100%	100%	100%

TAB 5. Influence du taux de mutation (P_m)

P_m	TPR	FPR	Precision	Accuracy	TNR
0	0	0	-	92%	100%
0.002	100%	0%	100%	100%	100%
0.005	100%	0%	100%	100%	100%
0.006	100%	0%	100%	100%	100%

Nous observons dans le tableau TAB 6 que le paramètre d'élitisme n'a pas d'influence sur les performances de détection (on retient alors L=2).. De même, le tableau TAB 7 indique que le nombre d'attaques injectées n'a pas d'influence sur la détection (TPR=100% et FPR=0%).

TAB 6. Influence de l'élitisme (L)

l	TPR	FPR	Precision	Accuracy	TNR
0	0	0	-	92%	100%
0.002	100%	0%	100%	100%	100%
0.005	100%	0%	100%	100%	100%
0.006	100%	0%	100%	100%	100%

TAB 7. Influence du nombre d'attaques injectées (I_a).

i_a	TPR	FPR	Precision	Accuracy	TNR
0	-	0%	-	100%	100%
2	100%	0%	100%	100%	100%
12	100%	0%	100%	100%	100%
24	100%	0%	100%	100%	100%

En résumé, pour différentes valeurs des paramètres Pm (taux de mutation) et L (nombre d'élites), toutes les attaques ont été détectées, indépendamment du nombre d'attaques injectées dans le fichier O. Pour évaluer les temps d'exécution, nous considérons les valeurs des paramètres suivantes :

P=50, Pm=0.005, L=2 et I_a=24. Le tableau TAB 8 indique les résultats obtenus pour le nombre de générations N : $1 \leq N \leq 220$

TAB 8. Evaluation des temps de détection

Nombre de générations (N)	Temps de détection (sec)	Nombre d'attaques détectées	TPR %	FPR %	Accuracy %	Precision %
1	0.069	17	70.83	0	70	100
5	0.014	19	79.16	0	79	100
50	**0.109**	**24**	**100**	**0**	**100**	**100**
100	0.216	24	100	0	100	100
200	0.564	24	100	0	100	100
220	0.660	24	100	0	100	100

Nous remarquons qu'il ya convergence de la méthode à partir de la génération 50. Toutes les attaques injectées, soit 24 attaques, sont donc détectées à la génération 50, après un temps de 0.109s. On a TPR=100% et FPR=0%. Ces résultats sont meilleurs que ceux obtenus avec AG (TPR = 99.5% et FPR= 0.43%).

3.4.3.2 Données aléatoires

Les résultats des tests suivants concernent une matrice Attaques-Evènements de dimension (400x200) générée aléatoirement. Les tests sont réalisés en utilisant les valeurs des paramètres par défaut de BBO donnés dans [Simon, 2008] (P=50, Pm =0.005, l=2), et 20 attaques injectées. Figure 3.6 montre l'évolution de TPR et FPR en fonction du nombre de générations (i.e. en fonction du temps). Nous observons qu'il existe une bonne discrimination entre les attaques présentes et les attaques absentes. En effet, dans le processus de détection, la population initiale est générée aléatoirement, aussi pour la génération 0, on a TPR ≈ 0.5 et FPR ≈ 0.5, et nous devons avoir après un certain nombre de générations toutes les attaques présentes détectées et aucune attaque absente détectée (TPR = 1 and FPR=0).

Remarquons que le nombre d'attaques injectées n'a aucune influence sur ces résultats.

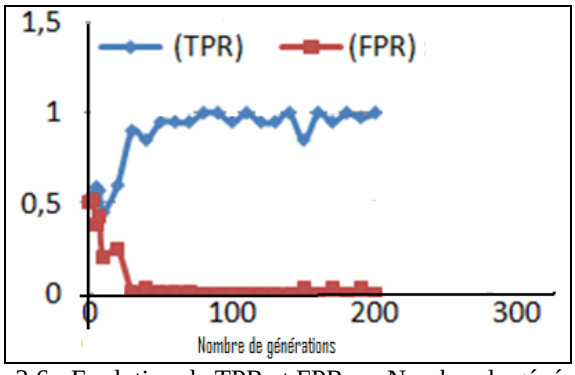


Figure 3.6 - Evolution de TPR et FPR vs. Nombre de générations

Dans le but d'apprécier la convergence de la fonction fitness ainsi que le temps nécessaire à la détection, nous calculons pour chaque génération le minimum, le maximum et la moyenne des valeurs de la fitness. La figure 3.7 montre que le minimum moyen, le maximum moyen et la moyenne de la fitness convergent rapidement vers l'optimum (après 50 générations).Le nombre d'attaques injectées n'a aucune influence sur les résultats. Cependant, il semble clair que le nombre de générations doit croître pour que la qualité de la détection soit de même qualité.

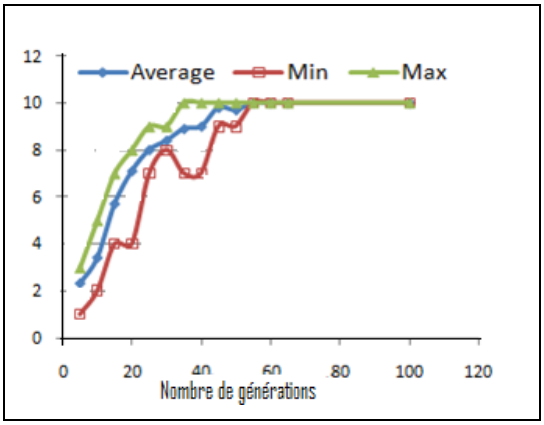


Figure 3.7 – Fitness moyenne, min et max pour 10 exécutions vs. Nombre de générations

Nous observons dans les figures 3.8 et 3.9, que la qualité de la détection se stabilise (TPR=1 et FPR=1) quand la taille de la population augmente (P≥80).

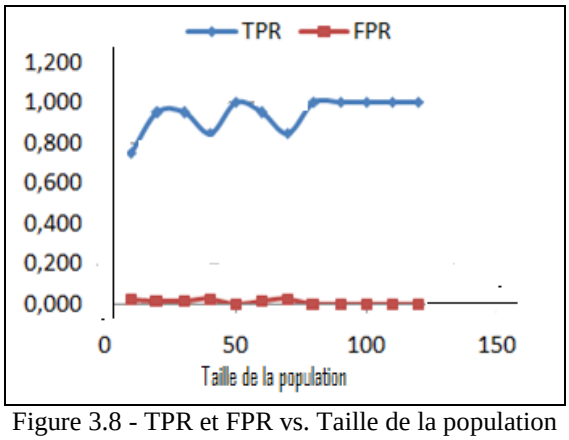


Figure 3.8 - TPR et FPR vs. Taille de la population

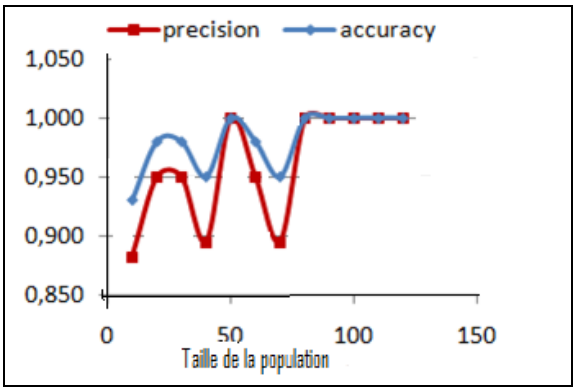


Figure 3.9 – Précision et Accuracy vs Taille de la Population

Figures 3.10 et Figure 3.11 montrent l'influence du taux de mutation P_m sur la qualité de la détection. De mauvais résultats sont obtenus pour Pm ≥ 0.01. Cependant, nous notons que le systèmene détecte pas les attaques quand le processus de mutation n'est pas utilisé. Le processus de mutation est important pour améliorer la performance du système.

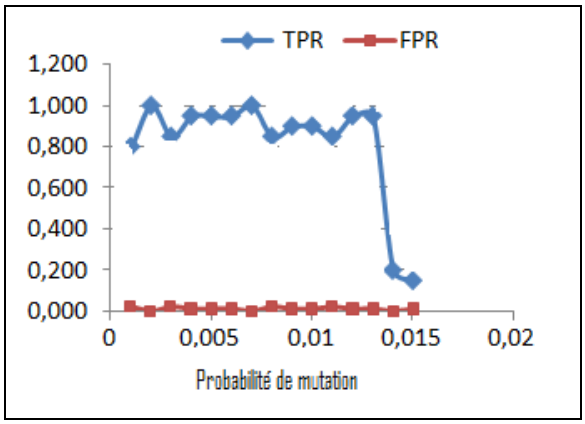


Figure 3.10 - TPR et FPR vs. Probabilité de mutation

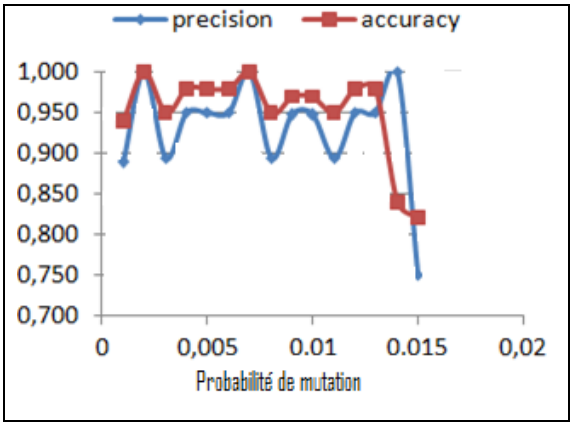


Figure 3.11 – Précision et Accuracy vs. Probabilitéde mutation

Figure 3.12 montre que la valeur du paramètre d'élitisme n'a pas une grande influence sur la qualité de la détection.

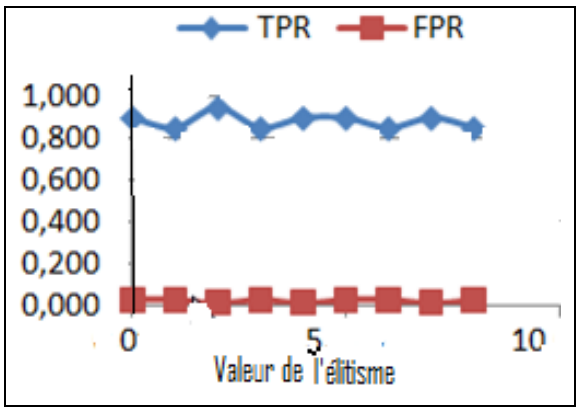


Figure 3.12 - TPR and FPR vs. Valeur d'élitisme

Pour différentes valeurs des paramètres P_m (taux de mutation) et L (nombre d'élites), toutes les attaques ont été détectées, indépendamment du nombre d'attaques injectées dans le fichier O.

Le tableau TAB 9 présente les résultats de tests effectués sur une matrice AE générée aléatoirement, de dimension 100 x300 (100 types d'évènements et 300 attaques). Les valeurs des paramètres utilisés sont : Taille de la population = 50 ; Taux de mutation = 0.005 ; valeur élitisme =2 et N (Nombre d'attaques injectées) =300.

TAB 9. Evaluation des temps de détection

Nb d'attaques détectées	Temps de détection (sec)
168	0.110
180	0.280
196	0.518
210	1.020
237	1.531
274	2.232
290	4.060
300	**9.040**

Toutes les 300 attaques sont détectées après un temps de 9.040 secondes.

La méthode de détection, basée sur la métaheuristique BBO, s'avère efficace et fiable. Les résultats montrent en effet les bonnes performances de la méthode. Un résultat important est la consistance des résultats, et cela indépendamment du nombre d'attaques réellement présentes dans le fichier d'audit analysé. Ce qui signifie que si il ya plusieurs attaques, la performance du système de détection n'est pas détériorée. Le temps d'exécution est très satisfaisant.

3.5 Analyse des traces d'audit de sécurité basée sur Harmony Search

Harmony Search (HS), est un algorithme évolutionnaire introduit en 2001[Geem et al., 2001]. Il a prouvé ses grandes capacités dans la résolution de différents problèmes d'optimisation [Coelho et Bernert, 2008], [Das et al, 2010]. La métaheuristique Harmony Search a été présentée au chapitre 1. Il s'agit ici de définir HS comme moteur de détection d'intrusions, au même titre que BBO, et d'évaluer ses performances.

3.5.1 Adaptation de Harmony Search

La méthode HS repose sur une optimisation stochastique et sur le processus de performance musical qui consiste à trouver l'harmonie parfaite dans un orchestre musical où chaque musicien joue une note pour trouver une meilleure harmonie. Nous sommes ici dans une situation où le codage des harmonies est immédiat puisque la solution du problème à résoudre s'exprime précisément sous la forme d'une suite binaire. Une harmonie est donc une suite de n_a notes pouvant prendre les valeurs 0 ou 1. Chaque harmonie correspond à une occurrence particulière du vecteur H (la somme des éléments le composant indique le nombre d'attaques détectées). En d'autres termes, la note i de l'harmonie vaudra 1 si l'harmonie correspond à une solution dans laquelle l'attaque i est déclarée présente. Dans le cas contraire, cette note prendra la valeur 0.

Le résultat que devrait retourner la méthode HS est un vecteur H sous la forme de suite binaire, où la valeur 1 indique la présence de l'attaque i dans le fichier d'audit O et la valeur 0 son absence. De plus, notre objectif étant de résoudre un problème de maximisation, la meilleure solution est associée à l'hypothèse H de plus grande valeur de la fonction $F=\sum R_i H_i$; $i \in \{0, 1,\ldots, n_a\}$ représentant la somme des risques encourus par le système sous surveillance. De plus, comme le problème est contraint, pour représenter une solution possible au problème, toute solution doit en effet respecter les inégalités : $(AE\times H)_i \leq O_i$ avec $0<i<n_e$. La prise en compte de ces contraintes consiste alors à éliminer les solutions qui ne les respectent pas, en fixant à zéro la valeur de la fonction objectif correspondante (ce qui représenterait pour HS une harmonie nulle, c'est-à-dire une harmonie où chaque note a une valeur zéro). Le processus de recombinaison se répétait jusqu'à ce qu'une solution vérifiant les contraintes soit générée. Rappelons que R_i représente le poids de l'attaque i, c'est-à-dire le risque encouru par le système si l'attaque n'était pas détectée.

Il y a lieu maintenant de représenter correctement la matrice HM. Cette dernière est une matrice de dimension ($HMS*n_a$), où n_a est le nombre d'attaques et HMS la taille de la population. A chaque ligne i de la matrice HM définissant une solution de notre problème, est associée une valeur de la fonction objectif F.

Les différents paramètres intervenant dans HS doivent être initialisés. Il s'agit de HMS, HMCR: Harmony memory consideration rate (taux d'utilisation de la mémoire), PAR : Pitch adjustment rate (taux d'ajustement). Le critère d'arrêt utilisé ici est le nombre maximum d'itérations.

L'étape suivante consiste en la génération des solutions initiales. Dans cette étape, HMS solutions sont générées aléatoirement, composant ainsi la matrice notée HM. La fitness F(i) correspondant à la solution i, *i*=1,.., HMS, est évaluée. L'évolution du processus d'optimisation consiste alors à générer une nouvelle solution $\mathbf{x} = (x_1, \ldots, x_{na})$ à partir de la matrice HM en utilisant les paramètres HMCR et PAR. Ces paramètres aident l'algorithme à obtenir des solutions améliorées localement ou globalement.

Cependant, la méthode HS standard, telle que décrite dans le chapitre 1, n'est pas totalement appropriée dans le traitement de problèmes discrets binaires, comme c'est le cas de notre problème d'analyse des traces d'audit de sécurité. Une extension de HS à la résolution efficace de problèmes binaires proposée dans [Wang et al., 2007] est alors utilisée.

Les règles de construction de la nouvelle solution (HMCR et PAR) sont modifiées. La solution est construite de la façon suivante :

Pour chaque variable x_k ($k = 1, \ldots, n_a$) :

$$x_k = \begin{cases} HM_{tk} \ , t \in \{1,\ldots\ldots HMS\} & \text{si } r_1 < HMCR \\ R & \text{sinon} \end{cases}$$

où :

$$R = \begin{cases} 0 & \text{si } r_2 < 0.5 \\ 1 & \text{sinon} \end{cases}$$

et r_1, r_2 sont deux nombres aléatoires indépendants compris entre 0 et 1 ; t est un nombre entier choisi aléatoirement dans {1,……, HMS}. L'opérateur d'ajustement est utilisé pour trouver localement de meilleures solutions. Dans le cas de problèmes binaires, il n'est pas surprenent d'obtenir des résultats limités avec HS standard. Pour y palier, une nouvelle régle d'ajustement est proposée dans [Wang et al. 2007].

Le paramètre PAR est utilisé après qu'une nouvelle solution $x = (x_1, \ldots, x_k, \ldots, x_{na})$ est construite et peut être appliqué à chaque variable de la solution. Dans ce cas, la valeur de chaque variable est modifié avec une probabilité PAR comme suite :

$$x_k = \begin{cases} HM_{bk} & \text{si } r < PAR \\ x_k & \text{sinon} \end{cases}$$

où r est nombre aléatoire entre 0 et 1; HM_{bk} représente la valeur de l'élément correspondant dans le vecteur de l'optimum global de HS.

Algorithme 14: Pseudo-code du processus d'analyse des traces d'audit de sécurité avec HS

Entrée : matrice AE et vecteur O
Initialiser des paramètres N, HMS, PAR et HMCR
Générer aléatoirement une mémoire d'harmonie (HMS solutions initiales)
Tester la validité des solutions générées
Evaluer la fitness des solutions générées
Nbgénérations =0
Tant que (Nbgénérations < max_Générations) **Faire**
  Construire la nouvelle solution **x** = (x_1, …, x_k ,…, x_{na})
  Evaluer la fitness de **x**
  Trouver la mauvaise harmonie
  Mettre à jour la mémoire d'harmonie
  Enregistrer la meilleur Harmonie dans le vecteur (H)
  Nbgénérations= Nbgénérations + 1
Fin Tant que
Sortie: Le meilleur vecteur H qui maximise le produit RxH et vérifie les contraintes.

Ainsi, si le nouveau vecteur d'harmonie x = (x_1, …, x_k ,…, x_{na}) est meilleur que le plus mauvais vecteur d'harmonie dans HM, en terme de valeur de la fonction objectif, la nouvelle harmonie est incluse dans HM et la plus mauvaise harmonie existante est exclue de HM.

L'algorithme 14 présente un pseudo-code de l'algorithme de résolution du problème d'audit de sécurité.

3.5.2 Tests et résultats

La procédure d'évaluation est identique à celle utilisée pour évaluer la méthode de détection avec BBO. Les tests sont réalisés d'abord sur le benchmark tiré de [Mé, 1998], puis sur des données générées aléatoirement. Durant les simulations, l'ensemble des attaques réellement présentes dans le fichier d'audit analysé doit être connu en avance. Ainsi, les évènements correspondant à une ou

plusieurs attaques sont inclus dans le vecteur audit observé. Chacune des expériences réalisée se caractérise par le 5–tuple (N, HMS, HMCR, PAR, Ia), où : N représente le nombre de générations, HMS la taille de la population (ou de la mémoire harmonie HM), HMCR le taux de considération de la mémoire harmonie, PAR le taux d'ajustement et I_a le nombre d'attaques réellement présentes dans le fichier d'audit (nombre d'attaques injectées). Les ratios TPR, FNR, accuracy et précision, définis précédemment, sont calculés pour évaluer la qualité de la détection. Tous les résultats ci-dessous sont obtenus comme la moyenne de 10 exécutions réalisées pour une même valeur du 5-tuple.

3.5.2.1 Benchmark

Les résultats des tests obtenus sur la matrice AE décrite dans [Mé, 1995] sont reportés dans les tableaux TAB 10, TAB 11 et TAB 12 ci-dessous.

- Le tableau TAB 10 présente l'influence du nombre de générations sur la qualité de détection. Les tests ont été réalisés avec : HMS = 30, HMCR = 0.98, PAR = 0.3 et Ia =20

TAB 10. Influence du nombre de générations

Nombre de générations (N)	Temps d'exécution (sec)	TPR %	FPR %	Accuracy %	Precision %
10	0.007	71	0	75.83	100
20	0.011	75	0	79.16	100
50	0.023	80	0	83	100
100	0.047	95	0	95.83	100
150	0.064	100	0	100	100
200	0.072	100	0	100	100

Nous avons fait varier le nombre de générations entre 0 et 200. Nous observons après un certain nombre de générations (N=150), que toutes les attaques présentes sont détectées et aucune fausse attaque (TPR = 1 and FPR=0). De plus le nombre d'attaques injectées n'a aucune influence sur ces résultats.

- Le tableau TAB 11 ci-dessous présente l'influence du paramètre HMS sur la qualité de la détection. Les tests ont été réalisés avec : N = 150, HMCR = 0.98, PAR = 0.3 et Ia=20. Nous avons fait varier HMS de 5 à 100 harmonies. Nous observons que les meilleurs résultats sont obtenus avec HMS $\in$ [20,30].

TAB 11. Influence de HMS

HMS	Temps d'exécution (sec)	TPR %	FPR %	Accuracy %	Precision %
5	0.073	95	0	95.8	100
10	0.053	95	0	95.8	100
20	0.053	100	0	100	100
30	0.058	100	0	100	100
50	0.054	95	0	95.83	100
70	0.055	90	0	91.66	100
100	0.048	90	0	91.66	100

- L'étude de l'influence des deux taux HMCR et PAR est important car ils contribuent fortement dans l'algorithme dans la recherche de la meilleur solution. Il ya une corrélation forte entre HMCR et PAR sur la processus d'optimisation de la métaheuristique Harmony Search [Wang et al., 2007].

TAB 12. Influence des taux PAR et HMCR

PAR	HMCR	Temps d'exécution (sec)	TPR %	Accuracy %
0.1	0.3	0.047	80	83.33
	0.5	0.048	85	87.5
	0.7	0.050	90	91.67
	0.8	0.049	95	95.83
	0.9	0.049	95	95.83
	0.98	0.047	100	100
0.3	0.3	0.048	80	83.3
	0.5	0.049	85	87.5
	0.7	0.052	90	91.67
	0.8	0.048	90	91.67
	0.9	0.049	95	95.83
	0.98	0.051	100	100
0.5	0.3	0.063	85	87.5
	0.5	0.050	85	87.5
	0.7	0.050	90	91.67
	0.8	0.047	95	95.83
	0.9	0.050	95	95.83
	0.98	0.048	100	100
0.7	0.3	0.049	80	83.33
	0.5	0.047	80	83.33
	0.7	0.049	95	95.83
	0.8	0.050	95	95.83
	0.9	0.050	95	95.83
	0.98	0.050	100	100
0.9	0.3	0.047	85	87.5
	0.5	0.048	85	87.5
	0.7	0.051	90	91.67
	0.8	0.049	100	100
	0.9	0.047	100	100

Pour étudier leur influence sur la qualité de la détection d'intrusions, nous avons fait varier le paramètre PAR entre 0.1 et 0.9 avec un pas de 0.2; Quant à HMCR, on le choisit dans l'ensemble {0, 0.3, 0.5, 0.7, 0.9, 0.98}. Les résultats sont présentés dans le tableau TAB 11. Ils ont été obtenus avec les valeurs des paramètres suivantes : N = 150, HMS = 30, Ia = 20.

Nous observons que les meilleurs résultats sont obtenus pour les valeurs de HMCR et PAR telles que : $0.8 \leq HMCR \leq 0.98$ et $PAR \geq 0.3$.

Le tableau TAB 13 indique les résultats de détection d'intrusion avec les valeurs des paramètres suivants : HMS=30, HMCR= 0.98, PAR= 0.3 et N, (nombre d'attaques injectées =24).

TAB 13. Tests de l'approche basée HS sur le benchmark

Nombre de générations (N)	Temps d'exécution (sec)	Nombre d'attaques détectées	TPR %	FPR %	Exactitude %	Précision %
1	0.002	16	66.67	0	66.67	100
5	0.003	179	70.83	0	70.83	100
50	0.020	20	83.83	0	83.83	100
100	0.037	22	91.67	0	91.67	100
200	**0.056**	**24**	**100**	**0**	**100**	**100**
220	0.064	24	100	0	100	100

Nous observons que la totalité des attaques injectées a été détectée après 0.056 seconde, devançant ainsi l'approche basée BBO qui a nécessité 1.02 seconde pour détecter les 24 attaques (TAB 7).

3.5.2.2 Données aléatoires

Nous considérons maintenant des matrices Attaques – Evènements (AE) de dimension plus importante, générées aléatoirement. De la même façon que précédemment, nous indiquons dans des tableaux TAB 14, TAB 15 et TAB 16, l'influence des différents paramètres de Harmony Search sur la qualité de détection d'intrusions. Les résultats portent sur une matrice AE générée aléatoirement de 200 attaques et 100 événements.

- Le tableau TAB 14 présente l'influence de la taille de la mémoire Harmonie (HMS) sur la qualité de la détection. Les tests ont été réalisés avec le nombre de générations fixé à 700, le taux de considération de la mémoire d'harmonie HMCR à 0.98 et le taux d'ajustement PAR à 0.3. Le nombre d'attaques injectées est 100. Nous avons fait varier HMS de 5 à 100 harmonies. Nous observons que les meilleurs résultats sont obtenus avec HMS entre 20 et 30.

TAB 14. Influence de HMS sur la qualité de la solution

HMS	Temps d'exécution (sec)	TPR %	FPR %	Accuracy %	Precision %
5	0.213	98	0	99	100
10	0.215	99	0	99.5	100
20	0.222	100	0	100	100
30	0.225	100	0	100	100
50	0.240	97	0	98.5	100
70	0.244	93	0	96.5	100
100	0.260	86	0	93	100

- Le tableau TAB 15 présente l'influence du nombre d'itérations sur la qualité de détection. Les tests ont été réalisés avec les paramètres : HMS = 30, HMCR = 0.98 et PAR à 0.3. Le nombre d'attaques injectées est 100. Nous avons fait varier le nombre de générations entre 0 et 750.

TAB 15. Influence du nombre de générations

Nombre d' itérations	Temps d'exécution (sec)	TPR %	FPR %	Exactitude %	Précision %
0	0.010	57	0	78.3	100
50	0.030	66	0	83	100
100	0.043	71	0	85.5	100
200	0.075	82	0	91	100
400	0.135	94	0	97	100
500	0.165	97	0	98.5	100
600	0.196	99	0	99.5	100
700	0.225	100	0	100	100
750	0.238	100	0	100	100

- Concernant l'étude de l'influence des paramètres PAR et HMCR sur la qualité de détection, nous avons fait varier le paramètre PAR entre 0.1 et 0.9 avec un pas de 0.2 ; Quant à HMCR, on le choisit dans l'ensemble {0.1, 0.3, 0.5, 0.7, 0.9, 0.98}. Les résultats présentés dans le tableau TAB 16 ont été obtenus en fixant : N = 700, HMS=30, et Ia = 100. Nous observons que les meilleurs résultats sont obtenus pour les valeurs de HMCR et PAR telles que : $0.8 \leq HMCR \leq 0.98$ et $PAR \geq 0.3$. Nous retenons les valeurs des paramètres suivants : HMS = 30, PAR = 0.3, HMCR = 0.98, et $N \geq 200$.

Cependant, il ya lieu de remarquer ici cette difficulté à paramétrer l'algorithme Harmony Search. De plus, il est hasardeux de conclure que la combinaison des quatre valeurs retenues, donnerait forcément la meilleure solution pour d'autres problèmes

TAB 16. Influence des taux PAR et HMCR

PAR	HMCR	Temps d'exécution (sec)	TPR %	Accuracy %
0.1	0.3	0.216	67	84
	0.5	0.212	71	85.5
	0.7	0.215	78	89
	0.8	0.214	84	92
	0.9	0.222	91	95.5
	0.98	0.219	99	99.5
0.3	0.3	0.210	69	84.5
	0.5	0 .220	73	86.5
	0.7	0.222	79	89.5
	0.8	0.220	83	91.5
	0.9	0.221	92	96
	0.98	0.227	100	100
0.5	0.3	0.211	69	84.5
	0.5	0.221	73	86.5
	0.7	0.218	78	89
	0.8	0.227	84	92
	0.9	0.232	92	96
	0.98	0.232	100	100
0.7	0.3	0.211	69	84.5
	0.5	0.218	75	87.5
	0.7	0.216	79	89.5
	0.8	0.221	83	91.5
	0.9	0.260	92	96
	0.98	0.229	100	10
0.9	0.3	0.210	69	84.5
	0.5	0.222	73	86.5
	0.7	0.222	77	88.5
	0.8	0.233	83	91.5
	0.9	0.235	93	96.5
	0.098	0.234	100	100

- Le tableau TAB 17 présente les résultats de tests effectués sur une matrice AE générée aléatoirement, de dimension 100 x300 (100 évènements et 300 attaques), que nous avons utilisée précédemment pour tester l'approche de détection basée sur BBO. Les valeurs des paramètres utilisées ici sont : HMS = 50, HMCR = 0.98, PAR = 0.3 et N (Nombre d'attaques injectées) =300.

Nous observons alors que toutes les attaques sont détectées après 3.773 secondes. Il apparaît clairement que les temps de détection sont nettement plus courts avec Harmony Search qu'avec BBO (résultats donnés dans TAB 9). Nous notons que ce résultat est conforme aux tests réalisés sur le benchmark.

TAB 17. Evaluation des temps de détection

Nb d'attaques détectées	Temps de détection (sec)
169	0.016
192	0.058
197	0.072
213	0.108
243	0.223
274	0.436
297	1.710
300	**3.773**

3.6 Conclusion

L'analyse des traces d'audit de sécurité peut être réalisée par la recherche d'attaques connues dans les fichiers des traces d'audit des activités de l'utilisateur. C'est un problème d'optimisation combinatoire NP-difficile. Les métaheuristiques offrent alors une alternative pour résoudre ce type de problème quand la taille des bases des évènements et des attaques croît. Dans les approches proposées, la métaheuristique Biogeography Based Optimization (BBO) et Harmony Search (HS) sont utilisées comme moteur d'analyse de systèmes de détection d'intrusions. BBO et HS sont deux algorithmes évolutionnaires à population de solutions bien conçus pour les problèmes d'optimisation sous contraintes. Les tests réalisés ont pour but le réglage des paramètres des métaheuristiques utilisées, ainsi que l'évaluation numérique des deux approches. Les tests furent réalisés sur des instances de grande taille générées aléatoirement ainsi que sur un benchmark tiré de [Mé,1998]. Des résultats expérimentaux de détection d'attaques simulées sont donnés. L'efficacité des deux approches est évaluée par leur capacité à faire des prévisions correctes. Elles se sont avérées efficaces et capables de produire deux méthodes fiables pour la détection d'intrusion. Les résultats montrent en effet les bonnes performances des deux approches proposées. Un résultat important fut la consistance des résultats, et cela indépendamment du nombre d'attaques réellement présentes dans le fichier d'audit analysé. Ce qui signifie que si il ya plusieurs attaques, la performance du système de détection n'est pas détériorée. Le temps d'exécution est très satisfaisant. Tout cela nous permet de considérer que les approches proposées constituent deux systèmes de détection d'intrusions efficaces et fiables.

Toutefois, les temps de détection des intrusions observés avec la méthode basée sur Harmony Search sont nettement inférieurs à ceux obtenus avec BBO.

Cependant, ces systèmes sont généralement développés pour des environnements bien définis et n'offrent pas une solution à certaines caractéristiques des réseaux telles que la variation des

comportements des utilisateurs et des services offerts, la complexité et l'évolution croissante des types d'attaques auxquels ils peuvent être sujets, la rapidité des attaques qui peuvent survenir simultanément sur plusieurs machines, etc.

Les métaheuristiques, en général, utilisent en outre un nombre plus ou moins important de paramètres qu'il s'agit d'évaluer avant toute utilisation, d'autant que le jeu de valeurs choisi a un impact considérable sur le temps fourni ainsi que la qualité des solutions. Souvent leur réglage se fait de façon empirique. Le thème du réglage des paramètres dans les métaheuristiques est abordé dans le chapitre suivant. La métaheuristique adaptative, sans paramètre de contrôle TRIBES, est présentée. TRIBES est inspirée de l'OEP, et développée pour la résolution de problèmes continus. Une extension des différents concepts utilisés dans TRIBES, à la résolution de problèmes discrets, et en particulier à la résolution du problème du PVC est développée.

Chapitre 4

Une métaheuristique sans paramètre pour la résolution de problèmes d'OC – Application au PVC

4.1 Introduction

L'utilisation des métaheuristiques pour la résolution de problèmes passe nécessairement par le réglage des paramètres, ce qui constitue un processus qui peut s'avérer long et difficile. Il est en effet important de trouver un jeu de paramètres bien adapté au problème posé, d'autant que chacun de ces paramètre a effectivement une forte influence sur le comportement de l'algorithme [Van den Bergh, 2002]. De ce fait, chaque problème nécessiterait en théorie une étude qui permettrait de dégager le jeu de paramètres optimal pour le traiter. Or, un tel procédé est souvent long à réaliser et demande une bonne connaissance au préalable du comportement de l'algorithme, ce qui le rend inapplicable dans un contexte industriel par exemple.

Dans le but de limiter l'intervention de l'utilisateur, de nombreuses études sur le choix des paramètres ont d'abord été mises en place au cours des dernières années. Cependant, bien que donnant une idée aux utilisateurs potentiels de l'influence des paramètres, ces études sont relativement restrictives, car portant souvent sur des intervalles limités de valeurs et ne sont applicables qu'à des topologies statiques de voisinages. Il existe aussi des études portant sur la définition des procédures automatiques de choix des paramètres [Birattari et al., 2002], [Adenso-Diaz et al.2006] . Ces procédures, bien qu'efficaces sur certains points, n'ont pour autant pas rencontré un vif succès. La principale raison est qu'elles ne s'appliquent qu'à un champ restreint de méthodes et elles ne se sont pas avérées robustes sur un large panel de problèmes. Enfin, la dernière façon de choisir les paramètres d'une méthode est justement de ne pas les choisir. Au lieu de définir des paramètres avant l'exécution de la méthode, on préfère ici modifier les valeurs des paramètres au cours du traitement, en fonction des résultats trouvés au cours des différentes itérations. De nombreuses métaheuristiques ont été conçues dans ce domaine : des algorithmes génétiques [Schnecke et al., 1996],[Murata et al., 2002], des algorithmes de colonies de fourmis [Chen et al., 2004],[Förster et al., 2007], des algorithmes de recherche tabou [Battiti, 1996] ou encore des algorithmes de recuit simulé [Ingber, 1996].

En 2003, Clerc [Clerc, 2003] a développé un algorithme adaptatif, un algorithme sans paramètres de contrôle inspiré de l'« Optimisation par Essaim Particulaires » (OEP), appelé TRIBES. La métaheuristique OEP a été décrite dans le chapitre précédent. L'évolution de l'essaim au cours du temps ainsi que le comportement des particules en constituent les deux phases essentielles de son

fonctionnement. La première est relative à la définition du paramètre N et de la topologie de voisinage. La deuxième est relative à la définition des paramètres w, c_1, c_2 et Vmax. Ainsi, pour définir TRIBES comme OEP adaptative, Clerc a défini deux types d'adaptations: les adaptations structurelles et les adaptations comportementales, qui seront décrites dans les sections suivantes.

Notons le caractère continu de TRIBES, dans le sens application à la résolution de problèmes continus. Une extension des concepts décrits dans TRIBES aux problèmes discrets est proposée dans la section 3.3.

4.2 Métaheuristique sans paramètre : TRIBES [Clerc, 2003]

Dans TRIBES, l'essaim de particules est divisé en plusieurs sous-essaims appelés tribus. Chaque tribu agit comme un essaim autonome, c'est à dire a son propre comportement global et explore une région particulière de l'espace de recherche. Toutes les tribus s'échangent des informations sur les régions qu'elles explorent. Comme le montre la figure 4.1, l'essaim est présenté comme un réseau interconnecté de tribus, qui sont elles-mêmes des réseaux interconnectés d'individus. Chaque particule est informée parelle-même (la mémoire cognitive), par tous les autres éléments de sa tribu, appelés informateurs internes (représentés par une ligne dans la figure 4.1) et si cette particule est la meilleure des particules de la tribu, appelée un chaman (représenté en noir dans la figure 4.1), alors elle est également informée par les chamans des autres tribus (appelé informateurs externes).

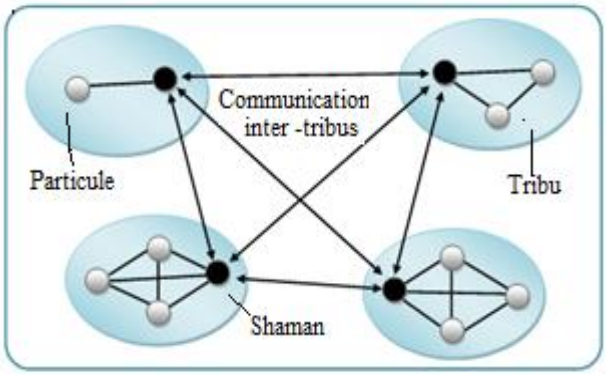


Figure 4.1 - Communications inter et intra tribus.

La mémoire sociale, notée G, d'une particule est l'informatrice avec la plus petite valeur de la fonction objectif. La topologie du graphe d'information de l'essaim est illustrée dans Figure 4.1. Les flèches symbolisent la communication inter-tribu alors que les traits matérialisent la communication intra-tribu. Les particules noires sont les chamans de chaque tribu. On verra dans la suite que cette structure est modifiée automatiquement par l'intermédiaire de créations et de destructions de particules. L'essaim doit être généré et modifié automatiquement, par le biais de la création, l'évolution

et la suppression de tribus sans avoir à définir de paramètres. Pour ce faire, des règles ont été introduites. Chaque particule a une position courante et une meilleure position. On définit alors des qualificatifs pour les particules et les tribus. Une particule est dite bonne ou neutre: bonne si elle vient d'améliorer sa meilleure performance, neutre, sinon. Quant à une tribu, elle est dite bonne, neutre ou mauvaise ; plus le nombre de bonnes particules dans la tribu est grand, plus elle est meilleure. Dans TRIBES, les particules sont rajoutées ou supprimées en fonction des comportements des tribus. Des adaptations structurelles et comportementales de l'essaim peuvent alors se produire.

4.2.1 Adaptations structurelles de l'essaim

Les mauvaises particules sont supprimées des bonnes tribus. La suppression d'une particule entraîne un changement dans le réseau de communications. Toutes les particules informatrices de la particule supprimée sont redirigées vers la meilleure particule de la tribu.

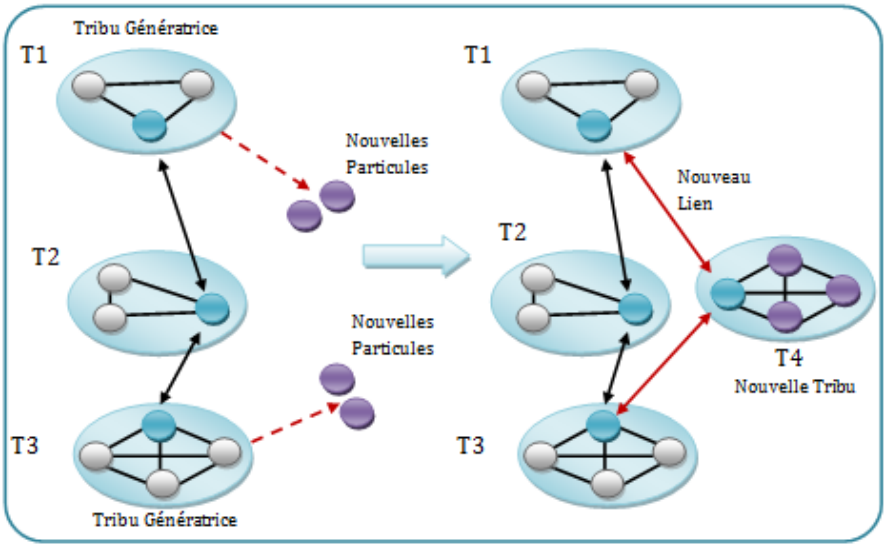


Figure 4.2 - Processus de génération

Les particules sont générées par les mauvaises tribus et forment une nouvelle tribu (Figure 4.2). Le nombre de particules générées par chaque mauvaise tribu est défini par l'équation ci-dessous, où D est la dimension de l'espace de recherche et tribeNb le nombre de tribus dans l'essaim.

$$NB_{particules} = Max\left(2, \left[\frac{9.5 + 0.124(D-1)}{TribeNb}\right]\right)$$

La mauvaise tribu gardera contact avec la nouvelle tribu et l'utilisera pour améliorer sa meilleure position.

Deux types de particules sont générés: les particules libres et les particules confinées, sachant que le type de particules générées est choisi aléatoirement. Nous précisons ci-dessous ces deux types de particules.

- **Les particules libres**: La particule est générée aléatoirement suivant une loi uniforme soit dans tout l'espace de recherche, soit sur sa frontière soit en un sommet de l'espace de recherche. Ce choix étant fait (aléatoirement), la particule est générée comme suit :

$$X_j = U\left(x_{min\,(j)}, x_{max\,(j)}\right), j \in \{1, \dots, D\} \qquad (1)$$

$$X_j = \begin{cases} U\left(x_{min\,(j)}, x_{max\,(j)}\right), j \in I \subset \{1, \dots, D\} \\ U\left(\{x_{min\,(j)}, x_{max\,(j)}\}\right), j \in J \subset \{1, \dots, D\} \end{cases} \qquad (2)$$

$$X_j = U\left(\{x_{\min\,(j)}, x_{\max\,(j)}\}\right), j \in \{1, \dots, D\} \qquad (3)$$

où $U(x_{min(j)}, x_{max(j)})$ est un nombre réel uniformément choisi dans $[x_{min(j)}, x_{max(j)}]$, et $U(\{x_{min(j)}, x_{max(j)}\})$ est un nombre réel uniformément choisi dans $\{x_{min(j)}, .., x_{max(j)}\}$. Les deux ensembles I et J sont définis aléatoirement pour chaque particule et déterminent une partition de $\{1,..,D\}$. La génération des particules libres vise à fournir une diversité dans la population.

- **Les particules confinées:** Elles visent à intensifier la recherche à l'intérieur d'un domaine de recherche prometteur. Soit $\vec{X}_i$ la meilleure particule de la tribu génératrice et $\overrightarrow{\iota_x}$ sa meilleure informatrice, et soient $\overrightarrow{P_x}$ and $\overrightarrow{P_{\iota_x}}$les meilleurs positions de $\vec{X}_i$ et $\overrightarrow{\iota_x}$. La nouvelle particule sera générée dans la D-sphère de centre center $\overrightarrow{P_{\iota_x}}$ et rayon $\left\|\overrightarrow{P_x} - \overrightarrow{P_{\iota_x}}\right\|$ telle que:

$$\vec{X}_{generated} = alea_{sphère}\left(\overrightarrow{P_{\iota_x}}, \left\|\overrightarrow{P_x} - \overrightarrow{P_{\iota_x}}\right\|\right)$$

où $alea_{sphère}\left(\overrightarrow{P_{\iota_x}}, \left\|\overrightarrow{P_x} - \overrightarrow{P_{\iota_x}}\right\|\right)$ est uniformément choisi dans l'hyper-sphère de centre $\overrightarrow{P_{\iota_x}}$ et rayon $\left\|\overrightarrow{P_x} - \overrightarrow{P_{\iota_x}}\right\|$.

Les adaptations structurelles ne doivent pas être effectuées à chaque itération. En pratique, si NL est le nombre de liens d'informations lors de la dernière adaptation, la prochaine adaptation sera effectuée NL/2 itérations plus tard. NL est estimé l'aide de l'équation :

$$NL = \sum_{n=1}^{tribe.Nb} explorerNb[n]^2 + tribeNb * (tribeNb - 1)$$

où tribeNb est le nombre de tribus et explorerNb[n] est le nombre de particules de la tribu n. Le premier terme du membre de droite de l'équation suppose que les particules d'une tribu sont complètement connectées. Donc, pour la tribu n, le nombre de liens d'information correspondant à la communication intra-tribu est égal à explorerNb[n]^2. Le second terme suppose quant à lui que tous les chamans sont reliés entre eux. Chaque chaman est relié à tous les autres, donc il possède (tribeNb – 1) liens d'informations correspondant à la communication inter-tribu. Les adaptations structurelles peuvent être résumées par l'Algorithme 15.

Algorithme 15 : Pseudo-code de l'algorithme Adaptations structurelles

```
Test = 0
Pour i=1 à Nb faire
   Si tribe(i)_status = bad
      Générer  NB_particules =  Max(2, [(9.5+0.124(D−1))/TribeNb]) particules
   Test = 1
   Fin Si
   Si tribe(i)_status = good
        Supprimer la plus mauvaise particule de tribe(i)
        Si tribe(i).size <> 1
            Rediriger les liens d'information vers la meilleur particule de la tribu
        Sinon
         tribesNb= tribesNb -1
         Rediriger les liens d'information vers la meilleure informatrice
         Fin Si
     Fin Si
 Fin Pour
Si Test=1
      tribesNb= tribesNb +1
     Agréger les particules générées à la  nouvelle tribu
      Etablir un lien avec les différents chamans
Fin Si
Calculer NL
```

Au début du traitement, l'essaim est composé d'une particule unique, qui constitue une tribu à elle seule. Si, durant la première itération, cette particule n'améliore pas sa performance, de nouvelles particules vont alors être générées pour former une deuxième tribu. NL=2 itérations plus tard, le même procédé est répété. On continue selon ce schéma durant toute l'exécution. La taille de l'essaim augmente jusqu'à ce que l'essaim trouve des zones prometteuses de l'espace de recherche. Plus la taille de l'essaim devient grande, plus le temps séparant deux adaptations va être long. La capacité d'exploration est améliorée et il va être plus facile pour les particules de trouver de bonnes solutions.

Une fois que des zones intéressantes de l'espace de recherche sont trouvées, les plus mauvaises particules vont être progressivement supprimées.

4.2.2 Adaptations comportementales de l'essaim

Le second moyen utilisé pour adapter l'essaim aux résultats obtenus par les particules est de choisir la stratégie de déplacement de chaque particule en fonction de son passé récent. Entre deux itérations, une particule peut améliorer sa performance, ce qui est notée (+), ou la détériorer, qui est désignée par (-). Il se peut qu'il n'y ait aucun changement aussi, à savoir un statu quo qui est désignée par (=). Ensuite, le choix de la stratégie de déplacement se fait selon les deux dernières variations comme indiqué ci-dessous:

Comportement	**Stratégie de déplacement**
(= +) (++)	Locale par par Gaussiennes independentes
(+ =) (- +)	Pivot Bruité
(- -) (= -) (+ -) (= =) (- =)	Pivot

- **La stratégie de pivot** est inspirée de travaux publiés, comme dans [13]. Soit$\vec{p}$ la meilleure position jamais atteinte par la particule et soit$\vec{g}$sa meilleure informatrice, *f* la fonction objectif. Le déplacement sera alors déterminé par:

$$\vec{X} = C_1 * aléa_{sphère}(H_p) + C_2 * aléa_{sphère}(H_g)$$

Où $C_1 = \frac{f(\vec{p})}{f(\vec{p})+f(\vec{g})}$,$C_2 = \frac{f(\vec{g})}{f(\vec{p})+f(\vec{g})}$,$aléa_{sphère}(H_p)$est uniformément choisidans l'hyper-sphère de centre$\vec{p}$et rayon$\|\vec{p} - \vec{g}\|$et $alea_{sphère}(H_g)$est uniformément choisi dans l'hyper-sphère de centre$\vec{g}$et rayon$\|\vec{p} - \vec{g}\|$.

- **La stratégie de pivot bruitée** est similaire à la stratégie précédente, avec en plus un bruit, tel que:

$$\vec{X} = C_1 * aléa_{sphère}(H_p) + C_2 * aléa_{sphère}(H_g)$$

$$b = \mathrm{N}\left(0, \frac{f(\vec{p}) - f(\vec{g})}{f(\vec{p}) + f(\vec{g})}\right)$$

$$\vec{X} = (1 + b)\vec{X}$$

En pratique, pour chacune des composantes de la dernière position calculée, un nombre aléatoire b est généré suivant une distribution gaussienne centrée réduite d'écart type $\frac{f(\vec{p})-f(\vec{g})}{f(\vec{p})+f(\vec{g})}$. La composante est alors multipliée par *(1+b)*.

- Dans la **stratégie locale par Gaussienne indépendante**, si $\vec{g}$= $(g_1,..g_D)$ est la meilleure informatrice de particule, alors le déplacement est déterminé par:

$$x_j = g_j + aléa_{normal}(g_j - x_j, \|g_j - x_j\|), j \in \{1, \dots, D\}$$

où, $aléa_{normal}(g_j - x, \|g_j - x_j\|), j \in \{1, \dots, D\}$est un point choisi aléatoirement suivant une distribution de Gauss de moyenne $(g_j - x_j)$ et d'écart type$\|g_j - x_j\|$.

L'algorithme 16 décrit TRIBES. g est la meilleure position atteinte par l'essaim, et p_i la meilleure position de la particule i. NL représente le nombre de liens de communication lors de la dernière adaptation de l'essaim, et n est le nombre d'itérations depuis la dernière adaptation de l'essaim. Concernant le critère d'arrêt, c'est généralement soit une erreur acceptable prédéfinie soit un nombre maximum raisonnable d'évaluations de la fonction objectif.

4.3 Adaptation de TRIBES à la résolution du PVC

4.3.1 Introduction

Plusieurs métaheuristiques d'origine combinatoire sont conçues pour être appliquées à des problèmes d'optimisation continus (les variables de décision sont continues) comme c'est le cas pour PSO et TRIBES. Par conséquent, nous revenons sur les différents mécanismes développés dans TRIBES, comme le processus de génération des particules ou les stratégies de déplacement développées dans les phases d'adaptation structurelle et comportementale. Une distance dans l'espace de recherche sera alors définie pour résoudre le POC, et en particulier le PVC. Des résultats d'implémentation indiquant les performances de notre algorithme seront fournis.

Algorithme 16 : Pseudo-code de l'algorithme TRIBES

Initialisation aléatoire d'une population de particules avec des positions
Evaluer la fonction objectif de chaque particule ; calculer g.
Pour tout individu, p_i est **initialisée à** X_i.
Tant que le critère d'arrêt n'est pas atteint **faire**
    Détermination du statut de chaque particule.
    Choix de la stratégie de déplacement
    Mise à jour des positions des particules.
    Evaluer la fonction objectif *f()* pour chaque individu.
    Si *n > NL*
        Détermination de la qualité des tribus
        Adaptation de l'essaim :
            Création et suppression de particules
            Restructuration du réseau de communication
        Calcul de *NL*
    Fin Si
Fin Tant que

4.3.2 Le problème du PVC

Le PVC [Bouillaguet, 2004] est l'un des problèmes les plus célèbres et largement étudié dans le domaine de l'optimisation combinatoire. Etant donné n villes et les distances les séparant deux à deux, le problème consiste à trouver un tour de longueur minimum qui visite les n villes une et une seule fois et se termine à la ville de départ. Il est généralement formulé en langage des graphes da la façon suivante: soit le graphe non orienté pondéré G=(S, E), où S est l'ensemble des sommets (villes), numérotés de 1 à n, avec |S|=n, et E l'ensemble des arêtes pondérées du graphe. Chaque élément de E est un triplé (i, j, w_{jj}), i ϵ{1, n}, j ϵ{1, n} et w_{ij} ϵ R. En outre, le graphe G=(S, E) est supposé être un graphe complet (les sommets sont adjacents deux à deux), si deux sommets ne sont pas adjacents, alors un est créé avec un poids assez grand pour être sûr qu'aucune solution ne peut contenir un tel arc virtuel. On rappelle qu'un cycle s sur k sommets est la donnée d'un k-uple (s_0, s_1, ..., s_{k-1}, s_k) tel que s_k=s_0 et (s_{i-1}, s_i) ϵ E. La longueur 'un tel cycle est la somme des longueurs des différentes arêtes qui le constituent. Un cycle qui passe par tous les sommets du graphe une et une seule fois est appelé cycle hamiltonien. Ainsi, le problème du PVC vise à déterminer un cycle hamiltonien de longueur minimum dans l'espace de recherche défini comme l'ensemble fini de tous les cycles hamiltoniens dans le graphe G=(S, E).

Lors de l'examen de TRIBES, un cycle hamiltonien s= (s_0, s_1, ..., s_{n-1}, s_n) avec s_n=s_0 et (s_{i-1}, s_i) ϵ E est ici considéré comme une «position». Une particule se déplace d'une solution à une autre, elle est

caractérisée par une matrice, qui change de valeur suivant sa position. De plus, on associe à toute position une valeur de la fonction objectif définie par :

$$f(\vec{X}) = \sum_0^n w_{i\,i+1}$$

Il est clair qu'elle a un nombre fini de valeurs, et son minimum global est la meilleure solution. Maintenant, nous allons définir sur l'espace de recherche une distance d de la façon suivante :

Soient h_1 et h_2 deux cycles hamiltoniens, et m le nombre d'arêtes communes à h_1 et h_2, alors: $d(h_1 , h_2) = n-m$. Les cycles h_1 et h_2 diffèrent d'un nombre d'arêtes égal à $d(h_1 , h_2)$ arêtes.

Après avoir précisé ces spécifications, nous pouvons présenter les différentes modifications apportées dans les mécanismes d'adaptation structurelle et comportementale définis dans TRIBES pour résoudre le problème du PVC.

4.3.3 Adaptations structurelles de l'essaim

Tel que défini dans TRIBES, initialement, l'essaim est un réseau de tribus de différentes tailles, qui sont elles-mêmes des réseaux inter connectés de particules. Lors des communications intra-tribus, chaque particule est informée par tous les autres éléments de sa tribu, et lors des communications inter-tribus, des liens sont créés, chaque nouvelle tribu garde le contact avec les tribus qui l'ont créée. Comme l'espace de recherche des solutions est discret, les processus de génération des particules libres et des particules confinées décrits dans TRIBES sont modifiés.

Les particules libres sont générées aléatoirement dans tout l'espace de recherche. Soit $\vec{X}=(x_1,..x_D)$ la position d'une particule. Chacune des composantes de $\vec{X}$ est choisie aléatoirement dans l'ensemble $\{0,\ldots,n-1\}$, de sorte à ne pas rajouter un sommet déjà existant dans $\vec{X}$.

Soit $\overrightarrow{x}$ la meilleure particule de la tribu génératrice et $\overrightarrow{\iota_x}$sa meilleure informatrice, et soit $\overrightarrow{P_x}$ et $\overrightarrow{P_{\iota_x}}$ les meilleures positions de $\vec{x}$ et$\overrightarrow{\iota_x}$ respectivement. Ainsi la particule confinée générée dans la *D*-sphere de centre $\overrightarrow{P_{\iota_x}}$ et rayon $\|\overrightarrow{P_x}-\overrightarrow{P_{\iota_x}}\|$ est:

$$\vec{X}_{\text{generated}} = \text{alea}_{\text{sphère}}\left(\overrightarrow{P_{\iota_x}} \;,\; \|\overrightarrow{P_x}-\overrightarrow{P_{\iota_x}}\|\right)$$

où $alea_{sphère}\left(\overrightarrow{P_{\iota_x}}, \left\|\overrightarrow{P_x}-\overrightarrow{P_{\iota_x}}\right\|\right)$ est uniformèment choisi dans l' hyper-sphere de centre $\overrightarrow{P_{\iota_x}}$ et rayon $\left\|\overrightarrow{P_x}-\overrightarrow{P_{\iota_x}}\right\|$, où $||\overrightarrow{P_x}-\overrightarrow{P_{\iota_x}}||$ est la distance entre les deux cycles hamiltoniens comme définie plus tôt..

L'idée est de générer aléatoirement un nombre entier *N* entre *0* and $R = \|\overrightarrow{P_x}-\overrightarrow{P_{\iota_x}}\|$, puis de procéder aléatoirement à *N* permutations dans $\overrightarrow{P_{\iota_x}}$.

Le cycle obtenu est alors la position courante de la nouvelle particule dans la sphère :

$$(\overrightarrow{P_{i_x}}, \|\overrightarrow{P_x} - \overrightarrow{P_{i_x}}\|)$$

4.3.4 Adaptations comportementales de l'essaim

Les différentes stratégies de déplacement développées dans TRIBES sont reprises et sont modifiées pour être appliquées à des problèmes discrets (ici pour le PVC).

4.3.4.1 La stratégie Pivot

La méthode pivot concerne les particules qui ont eu un mauvais comportement durant les deux dernières itérations. Le principe de la procédure donnée dans le cas continu est maintenu. Cependant, la nouvelle position est déterminée à partir de $\vec{p}$ et $\vec{g}$ en utilisant la distance **d** introduite plus tôt pour créer leurs voisinages pour remplacer les deux hyper-sphères H_p et H_g. Pour rappel, $\vec{p}$ est la meilleure position de la particule, $\vec{g}$ est sa meilleure informatrice, H_p et H_g sont deux hyper-sphères centrées en $\vec{p}$ et $\vec{g}$ respectivement et de rayon $\| \vec{p} - \vec{g} \|$.
La nouvelle procédure est :

i. Calculer la distance $\boldsymbol{d(\vec{p}, \vec{g})}$ comme rayon des hyper-sphères.

ii. Générer deux nombres aléatoires u_1 et u_2 uniformément distribués dans $(0, \boldsymbol{d(\vec{p}, \vec{g})})$ et procéder à u_1 permutations dans $\vec{p}$ et u_2 permutations dans $\vec{g}$ (deux points: $point_1$ et $point_2$ sont alors uniformément générés dans l'hyper-sphère H_p and H_g respectivement).

iii. Calculer les coefficients d'attraction $\boldsymbol{c_1} = \frac{f(\vec{p})}{f(\vec{p})+f(\vec{g})}$ et $\mathbf{c_2} = \frac{f(\vec{g})}{f(\vec{p})+f(\vec{g})}$

iv. Pour toute composante x_i de la dernière position calculée de la particule, générer un nombre aléatoire u_3 uniformément distribué dans $(0,1)$.

Si $u_3 > c_2$, alors la composante de $point_1$ est choisie

Si $u_3 < c_2$, alors la composante de $point_2$ est choisie

Les composantes de la nouvelle position sont les éléments choisis dans **(4)** et les composants doivent être différents deux à deux.

4.3.4.2 La stratégie Pivot Bruitée

La méthode pivot bruitée concerne les particules de moyenne performance dans les deux dernières itérations. La procédure proposée préserve la méthode pivot bruitée originale.

La nouvelle procédure est :

i. Générer la nouvelle position $\vec{X}'$ en utilisant la méthode pivot adaptée, ci-dessus.

ii. Générer un nombre aléatoire b suivant une loi normale centrée réduite d'écart type $\frac{f(\vec{p})-f(\vec{g})}{f(\vec{p})+f(\vec{g})}$.

iii. Calculer k=| n.b. |

iv. Procéder à k permutations dans la dernière position de la particle pour obtenir la nouvelle position.

4.3.4.3 La stratégie Locale avec Gaussiennes indépendantes

Dans cette stratégie, le principe est d'intensifier la recherche autour de la meilleure informatrice $\vec{g}$. Aussi le voisinage de $\vec{g}$ sera déterminé en utilisant la distance d définie plus tôt. Puis, un voisin est généré suivant une loi normale comme suit:

i. Calculer la distance d ($X^{\rightarrow}$,$g^{\rightarrow}$), $X^{\rightarrow}$ est la position courante.

ii. Générer un nombre aléatoire entier k suivant une loi normale centrée réduite, d'écart type d ($\vec{X}$,$\vec{g}$). Procéder à k permutations dans la dernière position de la particule pour obtenir la nouvelle position.

Remarque: Les étapes de TRIBES présentées plus tôt dans ce chapitre et non redéfinies dans cette section restent inchangées.

4.3.5 Résultats expérimentaux

Plusieurs tests sont réalisés sur des données générées aléatoirement. Nous avons observé le bon comportement de TRIBES comme le montrent les graphes de la Figure 4.3 qui indiquent les valeurs de la fonction objectif après 50 000 itérations pour des instances de 200 et 400 villes respectivement. L'axe des abscisses représente le numéro de l'itération où une modification de la valeur de la fonction objectif a eu lieu.

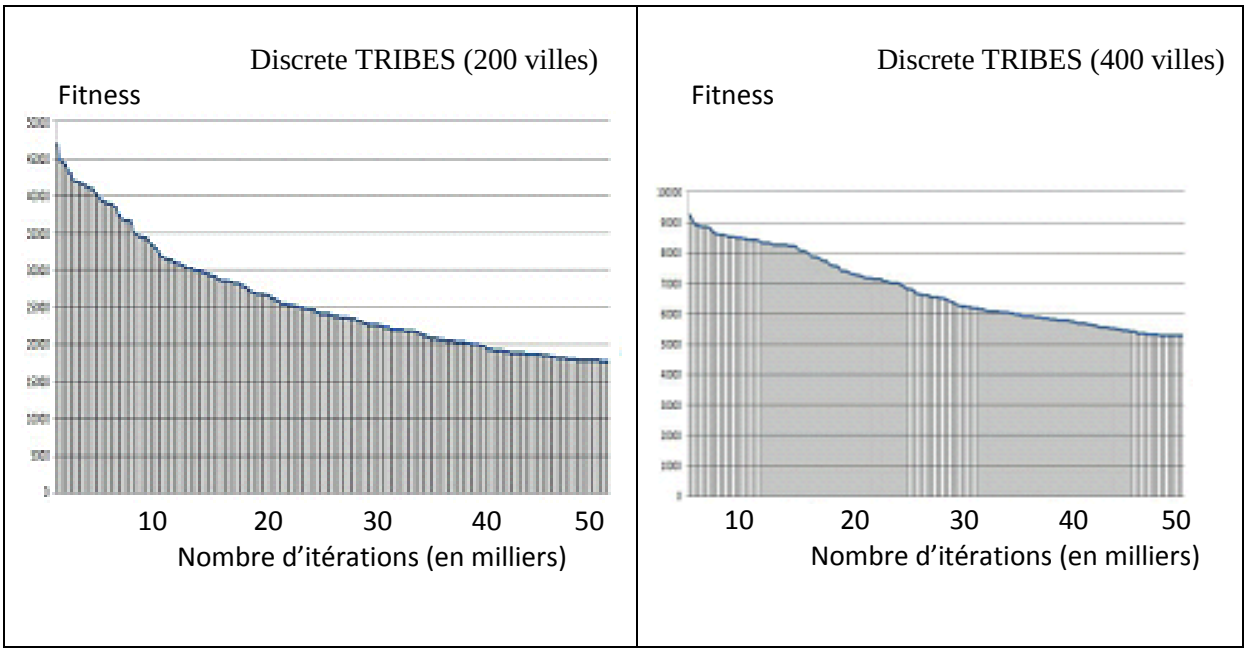


Figure 4.3 - Comportement de la fonction objectif Discrete TRIBES

Une comparaison avec les algorithmes génétiques (AG) a été réalisée. Les résultats affichés (voir Figure 4.4) sont obtenus comme moyenne de 10 exécutions différentes (50000 itérations). Plusieurs tests ont été réalisés avec les AG utilisant différents jeux de paramètres. La meilleure solution a été retenue avec le jeu de paramètres correspondant, donné dans le tableau TAB 1.

TAB 18. Paramètres de l'AG

Population initiale	Nombre de générations	Taux de mutation	Taux de croisement
1000	200 000	30 %	10 %

Pour les instances utilisées, les résultats, comme indiqués dans la Figure 3.4, sont très compétitifs.

Nous notons, cependant, que pour plusieurs instances issues de la librairie 'TSPLIB', les résultats obtenus restent mitigés : la méthode converge rapidement vers une solution dont la fitness reste assez loin de l'optimum attendu. Le problème rencontré est celui de la convergence prématurée. Le processus d'optimisation n'arrive pas à s'extraire des extrema locaux. Nous n'avons pas traité ce problème.

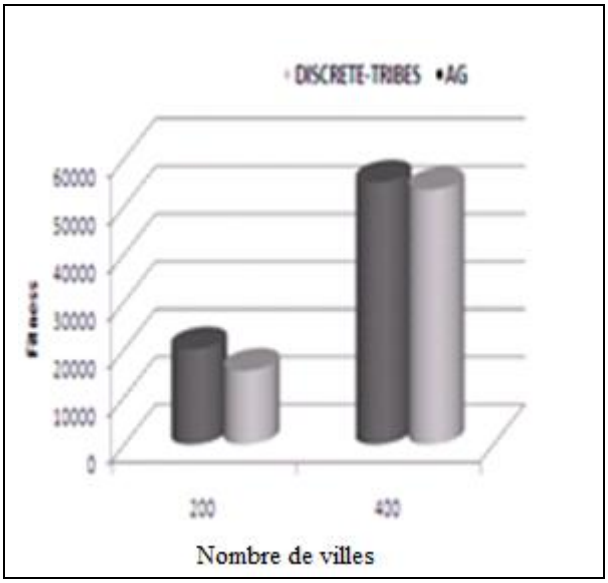


Figure 4.4 Dicrete TRIBES vs AG (200 et 400 villes)

Cependant, il est clairement précisé par les concepteurs de TRIBES dans [Coreen et al., 2008], que TRIBES présente un défaut majeur qui est la convergence prématurée. Des modifications ont alors été apportées, améliorant nettement les résultats. Ces modifications consistent en l'introduction de deux nouveaux mécanismes dans la méthode :

- Le premier a consisté en une initialisation régulière des particules de manière à ce que, l'espace de recherche soit exploré de la manière la plus large possible par l'essaim de particules tout au début du traitement. L'idée vient du fait que les performances des méthodes d'optimisation par essaims particulaire (OEP) sont directement liées à l'initialisation des particules [Campana et al., 2006] et que la trajectoire d'une particule est dépendante de sa position initiale et de sa vitesse initiale.

- Le deuxième mécanisme est la définition d'une nouvelle stratégie de déplacement pour améliorer le comportement des particules défaillantes. Ce deuxième mécanisme utilise un algorithme à estimation de distribution, qui a permis d'avoir des déplacements plus larges, conditionnés par les meilleures particules de l'essaim et, ainsi, de diversifier le comportement d'une particule défaillante.

4.4 Conclusion

Un problème important émerge lors de l'utilisation des métaheuristiques, à savoir le réglage des paramètres. Cette opération est coûteuse en termes de temps et requiert un savoir-faire et une expérience de la part de l'utilisateur. L'idée est donc de trouver une approche pouvant être utilisée comme étant une boite noire, délivrant des résultats proches de l'optimum en un temps raisonnable. Maurice Clerc a développé une telle méthode de résolution : TRIBES, dérivée de l'approche PSO. Cependant celle-ci étant destinée exclusivement aux problèmes continus, elle s'avère inadéquate pour

la résolution de problèmes de nature discrète. Une extension de TRIBES, appliquée à la résolution du problème du voyageur de commerce, a été présentée, avec un bon comportement de la fonction objectif. Une amélioration de TRIBES a été apportée dans [Coreen et Clerc, 2008]. Les deux nouveaux mécanismes introduits constituent une piste pour une amélioration de notre extension de TRIBES à la résolution du PVC.

CONCLUSION GENERALE

Dans cette thèse, nous avons présenté un travail qui a porté d'une part sur la contribution des métaheuristiques d'optimisation, de la famille des algorithmes évolutionnaires, à la résolution du problème de la détection d'intrusion par l'analyse des traces d'audit de sécurité, et d'autre part à la résolution du Problème du Voyageur de Commerce par l'utilisation d'une métaheuristique sans paramètre adaptée de TRIBES : algorithme d'Optimisation par Essaims Particulaires, adaptatif conçu pour résoudre des problèmes continus. Les problèmes étudiés sont NP-Difficiles.

L'analyse des traces d'audit de sécurité, est un mécanisme de détection d'intrusion à postériori, à base de signatures d'attaques. Il s'agit alors de rechercher dans le fichier des traces d'audit de sécurité les occurrences d'attaques. Ce problème est NP-Difficile, nécessitant alors l'utilisation de métaheuristiques pour sa résolution. Deux méthodes de détection dites à postériori sont proposées, utilisant les métaheuristiques : Biogeography Based Optimization (BBO) et Harmony Search (HS). Ces dernières constituent les moteurs d'analyse des systèmes de détection d'intrusions. Ce sont deux algorithmes évolutionnaires de la famille des techniques de l'intelligence artificielle, lesquelles, de par leurs caractéristiques, telles que l'adaptation, la vitesse de calcul élevée, répondent aux exigences de la construction d'un bon modèle de détection d'intrusion. Des tests ont été réalisés pour évaluer la qualité de détection, et pour analyser l'influence de chacun des paramètres de la métaheuristique sur la qualité de la détection. Une bonne discrimination entre les attaques présentes et les attaques absentes est relevée. Toutefois, les temps de détection des intrusions observés avec la méthode basée sur Harmony Search sont inférieurs à ceux obtenus avec BBO. Plusieurs perspectives sont envisagées, qui prendraient en considération la réalisation multiple d'une attaque particulière, des attaques indépendantes ou la chronologie des évènements.

Le réglage des paramètres des métaheuristiques constitue un problème majeur lors de leur utilisation. Des valeurs de nombreux paramètres doivent être déterminées, d'autant que ces paramètres peuvent avoir une grande influence sur l'efficience et l'efficacité de la recherche de la solution. Cette opération peut consommer beaucoup de temps et requiert un savoir-faire et une expérience de la part de l'utilisateur. Des algorithmes dits adaptatifs ont été proposés dans la littérature pour palier au problème du réglage des paramètres, avec plus ou moins de succès. Les valeurs des paramètres sont alors modifiées en fonction des résultats collectés durant le processus de recherche et non imposées par l'utilisateur. Maurice Clerc a proposé TRIBES, qui est un algorithme conçu pour la métaheuristique « Optimisation par essaims particulaires ». Il présente la particularité d'être totalement adaptatif : l'utilisateur n'a qu'à définir son problème et son critère d'arrêt, sans se préoccuper des paramètres de l'algorithme. Tous les paramètres de contrôle sont calculés de manière autonome par l'algorithme, aucune intervention humaine n'étant nécessaire à la bonne résolution d'un problème. Cependant cette méthode, étant destinée exclusivement aux problèmes continus, s'avère inadéquate

pour la résolution de problèmes de nature discrète. Une extension de TRIBES, adaptée à la résolution de problèmes d'optimisation combinatoire est alors présentée. Elle est appliquée à la résolution du problème du voyageur de commerce. Différents tests sont réalisés, certains affichant un bon comportement, comparables aux résultats d'un AG. Cependant, le problème de convergence prématurée s'est posé : l'algorithme converge rapidement vers un optimum, qui est souvent un optimum local, et n'arrive pas à s'en extraire, et cela à cause d'un manque de diversité dans l'exploration de l'espace de recherche. En d'autres termes, les mécanismes de l'algorithme n'arrivent pas à maintenir la diversité nécessaire parmi les particules, afin d'explorer les différents optima locaux de la fonction objectif. Nous n'avons pas traité ce problème. Il reste dans nos perspectives. Signalons toutefois, que le problème de convergence prématurée a été relevé lors de l'utilisation de TRIBES originelle pour les problèmes continus, des améliorations ont alors été proposées [Coreen et al., 2008].

REFERENCES BIBLIOGRAPHIQUES

[Aarts et al., 1997] Aarts, E.H.L., Korst, J.H.M. and Laarhoven, P.J.M. van (1997). Simulated annealing. In E.H.L. Aarts & J.K. Lenstra (Eds.), Local search in combinatorial optimization (pp. 91-120). Chichester: Wiley-Interscience

[Abadeh et al., 2007] Abadeh, S., Habibi, M., and Lucas, J. (2007). Intrusion Detection Using a Fuzzy Genetics-Based Learning Algorithm. Journal of Network and Computer Applications, pages 414-428.

[Adenso-Diaz et al., 2006] Adenso-Diaz, B., and Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental design and local search, Operations Researchs, Vol. 54, N°1, 2006.

[Ahmim et al., 2011] Ahmim, A., Ghoualmi, N., and Kahya, N. (2011). Improved Off-Line Intrusion Detection using a Genetic Algorithm and RMI. In: Proc. of the International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 2, N°.1.

[Al-Betar et al., 2010] Al-Betar, M., Khader, A., and Liao, I. (2010). A harmony search with multi-pitch adjusting rate for the university course timetabling. In: Geem Z (ed) Recent advances in Harmony search algorithm. Springer-Verlag, Berlin, Heidelberg, pp 147–161

[Alia et al., 2009] Alia, OM., Mandava, R., Ramachandram, D., and Aziz, ME. (2009). Dynamic fuzzy clustering using harmony search with application to image segmentation. In: IEEE international symposium on signal processing and information technology (ISSPIT09). pp 538–543

[Amor et Rettinger] Amor, H.B., and Rettinger, A. (2005). Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation. In: Genetic and EvolutionaryComputation Conference, GECCO, pp 1531–1538, doi:10.1145/1068009.1068250

[Anderson, 1980] Anderson, J. (1980). Computer Security Threat Monitoring and Surveillance. Technical Report 79F296400, James P. Anderson, Co., Fort Washington, PA.

[Bace, 2000] Bace, R. G. (2000). Intrusion Detection Systems. Mac Millan Technique Publication, USA.

[Bace et Mell, 2001] Bace,R., and Mell, P. (2001). NIST Special Publication on Intrusion Detection Systems.

[Bachelet, 1999] Bachelet. V. (1999). Métaheuristiques parallèles hybrids: Application au QAB. PhD thesis, USTL LIFL France.

[Back et al., 2000] Back, T., Fogel, D. and Michalewicz, Z. (2000). Evolutionary Computation 1: basic algorithms and operators. Institute of Physics.

[Back et al., 1997] Back, T., Fogel, D. and Michalewicz, Z editors. (1997). Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol, UK, 1st edition.

[Bachmann, 1894] Bachmann, P.G.H. (1894). Die Analytische Zahlentheorie / Dargestellt von

Paul Bachmann, Zahlentheorie: Versuch einer Gesamtdarstellung dieser Wissenschaft in ihren Haupttheilen, vol Zweiter Theil. B. G. Teubner, Leipzig, Germany.

[Balasubramaniyan et al., 1998] Balasubramaniyan, J.J., and Garcia-Fernandez, J.O.,Isaco, D., Spaord,E.H., and Zamboni, D. (1998). An architecture for intrusion detection using autonomousagents. In ACSAC, pages 13_24, 1998.

[Banković et al., 2009] Banković, Z., Álvaro, A., de Goyeneche, J.M (2009). Intrusion Detection in Sensor Networks Using Clustering And Immune systems. Lecture Notes in Computer Science, Volume 5788, Intelligent Data Engineering and Automated Learning - IDEAL 2009, pages 408-415.

[Banks et al., 2007] Banks, A., Vincent, J., Anyakoha, C. (2007). A review of Particle Swarm Optimization. Part I: background and development, Natural Computing, Vol. 6, N°4, pp. 467-484, 2

[Battiti, 1996] Battiti, R. (1996). _Reactive search: toward self tuning heuristics_, Modern Heuristic Search Methods, pp. 61-83, John Wiley & Sons.

[Biondi, 2001] Biondi, P. (2001). Architecture expérimentale pour la détection d'intrusions dans un système informatique.

[Birattari et al., 2002] Birattari,M., Stützle, T., Paquete, L.,Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics_, Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002, pp. 11-18, Morgan Kaufmann.

[Blum et Roli, 2003] Blum, C., and Roli, A., (2003). Metaheuristics in combinatorial optimization : overview and conceptual comparison. ACM Computing Surveys, 35:268–308, 2003.

[Bledsoe et Browning, 1959] Bledsoe, W.W., and Browning I. (1959). Pattern recognition and reading by machine. In: Proceedings of the Eastern Joint Computer Conference (EJCC) – Papers and Discussions Presented at the Joint IRE - AIEE - ACM Computer Conference, pp 225–232.

[Bouillaguet, 2004] Bouillaguet, C. (2004). Le Problème du voyageur de Commerce.

[Bremermann, 1962] Bremermann, H.J. (1962). Optimization through evolution and recombination. Self-Organizing systems pp 93–10.

[Brown, 1979] Browne, P. (1979) «The Security Audit ». Dans Checklist For Computer Secturity Center Self-Audits, pages 173-184.

[Burke et al.,2004] Burke, E.K., Gustafson, S.M., and Kendall G. (2004). Diversity in genetic programming: An analysis of measures and correlation with fitness. IEEE Transactions on EvolutionaryComputation8(1):47–62, doi:10.1109/TEVC. 2003. 819263.

[Campana et al.,2006] Campana, E. F., Fasano, G., Peri, D.,and Pinto, A. (2006). Particle swarm optimization: Efficient Globally Convergent Modifications. Proceedings of the 3rd Eurppean Conference on Computational Mechanics, Solids and Coupled Problems in Engineering, 2006, Springer.

[Cerf, 1994] Cerf, R. Une théorie asymptotique des algorithmes génétiques. PhD thesis, Université de Montpellier II, France, 1994.

[Cha et Sad, 2004] Cha S, and Sad. (2004). Web Session Anomaly Detection Based on

Parameter Estimation. Computers & Security, Vol.23, No.4, pages 265-351.

[Chen et al, 2004] Chen, X.H. Xu, Y.X. Chen. (2004). An Adaptive Ant Colony Clustering Algorithm, Proceedings of the 3rd Conference on Machine Learning and Cybernetics, 2004,pp. 1387-1392, IEEE Press.

[Clerc, 2003] Clerc, M. (2003). TRIBES - Un exemple d'optimisation par essaim particulaire sans paramère de contrôle_, Conférence OEP'03, 2 Octobre , 2003, Paris, France.

[Coelho et Bernert, 2008] Coelho L. S, and Andrade Bernert, D. L. (2008). An improved harmony search algorithm for synchronization of discrete-time chaotic systems. Industrial and Systems Engineering Program, LAS/PPGEPS, Pontifical Catholic University of Paraná, PUCPR, Imaculada Conceição, 1155, 80215-901 Curitiba, Paraná, Brazil.

[Colorni et al., 1992] Colorni, A., Dorigo, M., and ManiezzoV. (1992). Distributed Optimization by Ant Colonies, Proceedings of the 1rst European Conference on Artificial Life, 1992, pp. 134-142 Elsevier Publishing.

[Cook, 1971] Cook, S. (2003). The complexity of theorem proving procedures. Proceedings of the Third Annual ACM Symposium on Theory of Computing, pp. 151–158.

[Coreen et al., 2008]. Cooren, Y., Clerc,M., and Siarry, P.(2008). Initialization and Displacement of the Particles in TRIBES, a Parameter-Free Particle Swarm Optimization Algorithm. Adaptive and Multilevel Metaheuristics: 199-219

[Crosbie et Spafford, 1995] Crosbie, M., and Spafford, E.H. (1995) Applying genetic programming to intrusion detection, Proceedings of the 1995 AAAI Fall Symposium on Genetic Programming.

[Daoudi et al., 2011a] Daoudi, M., Boukra, A., and Ahmed--Nacer, M. (2011) Adapting TRIBES algorithm for Traveling Salesman Problem. Proceedings of the International Symposium on Programming and Systems: ISPS, and IEEEXPLORER

[Daoudi et al.,2011b] Daoudi, M., Boukra, A., and Ahmed-Nacer, M. (2011) Security Audit Trail Analysis with Biogeography Based Optimization Metaheuristic. In the Proceedings of the International Conference on Informatics Engineering & Information Science: ICIES 2011, A. Abd Manaf et al. (Eds.): ICIEIS 2011, Part II, CCIS 252, pp. 218–227, 2011.© Springer-Verlag Berlin Heidelberg 2011

[Daoudi et al.,2011c] Daoudi, M., Boukra, A., and Ahmed-Nacer, M. (2011) .A biogeography Inspired Approach for Security Audit Trail Analysis. In: Journal of Intelligent Computing Volume 2 Number 4 December 2011

[Darwin, 1859], Darwin, C. On the origin of species. (1859). John Murray, London.

[Das et al., 2010] Das, S., Mukhopadhyay, A., Roy A, Abraham, A., and Panigrahi, B. K. (2010). Exploratory Power of the Harmony Search Algorithm: Analysis and Improvements for Global Numerical Optimization. IEEE. Transactions on systems, man, and cybernetics—part b: cybernetics. 1083-4419/$26.00 © 2010 IEEE

[Dass, 2003] Dass, M. (2003). A Learning Intrusion Detection System. Master of Science, The University of Georgia, Athens, Georgia.

[Deb, 2001] Deb, K. (2001). Genetic algorithms for optimization. KanGAL Report 2001002, Kanpur Genetic Algorithms Laboratory (KanGAL), Kanpur, PIN 208 016, India ,

[Denning, 1987] Denning, D (1987). An Intrusion-Detection Model. In Proc. of the 1986 IEEE Symposium on Security and Privacy, pages 118-131.

[Dewan et al., 2009] Dewan, M. F., Darmont, J., Harbi, N., Huu Hoa, N., and Rahman, M. Z. (2009). "Adaptive Network Intrusion Detection Learning: Attribute Selection and Classification", In "International Conference on Computer Systems Engineering (ICCSE 2009), Bangkok : Thailand.

[Dokas et al., 2002] Dokas, P,. Ertoz, E., Kumar, V,. Lazarevic, A,. Srivastava, J,.and Pang-Ning Tan. Data mining for network intrusion detection. University of Minnesota, Minneapolis, MN 55455, USA, 2002.

[Dorigo et al., 1991] Dorigo, M., Maniezzo, V., and Colorni, A. (1991). Positive feedback as a search strategy. Technical Report 91-016, Politecnico di Milano, Italy, 1991.

[Dorigo, 1992] Dorigo, M. (1992). Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italy.

[Dorigo et Di Caro, 1999] Dorigo, M., and Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, et F. Glover, editors, New Ideas in Optimization, pages 11–32, London, 1999. McGraw Hill.

[Dorigo et al., 1999] Dorigo, M., Di Caro, G., and Gambardella, L. M. (1999). Ant algorithms for discrete optimization. Articial Life, 5 :137–172.

[Dorigo et al., 2002] Dorigo, M., Gambardella, L., Middendorf, M., and Stutzle, T. (2002). (eds.), Special section on ant colony optimization, IEEE Transactions on Evolutionary Computation (6), August 2002.

[Dorigo et al., 2004] Dorigo, M and Stutzle, T. (2004). Ant Colony Optimization (The MIT Press, 2004).

[Dréo et al., 2006] Dréo, J., Pétrowski, A., Siarry, P., and Taillard, É. (2006). Metaheuristics for Hard Optimization. Springer-Verlag.

[DuMouchel, 1999] DuMouchel, W. (1999). Computer intrusion detection based on Bayes factors for comparing command transition probabilities. Technical Report TR91, National Institute of Statistical Sciences (NISS).

[Eberhart et al., 2001] Eberhart, R, Shi, Y., and Kennedy, J. (2001). Swarm Intelligence (Morgan Kaufmann, 2001).

[Eberhart et Shi., 2004] Eberhart, R and Shi, Y. (2004) (eds.), Special issue on particle swarm optimizaion, IEEE Transactions on Evolutionary Com,.putation (8).

[Eiben et Smith, 2003] Eiben, A. E. and Smih, J. E. (2003). Introduction to Evolutionary Computing, Springer.

[Eiben et Schippers, 1998] Eiben, A.E., and Schippers, C.A. (1998). On evolutionary exploration and exploitation. Fundamenta Informaticae 35(1-4):35–50

[Eshelman et al., 1989] Eshelman, L.J., Caruana, R.A., and Schaffer, J.D. (1989). Biases in the

crossover landscape. In:Proceedings of the third international conference on Genetic algorithms, pp 10–19.

[Evangelista, 2004] Evangelista, T. (2004). « Les IDS Les système de détection d'intrusion informatique ». Edition DUNOD, 2004.

[Feo et Resende, 1995] Feo, T. A., and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. Journal of Global Optimization, 6, pp.109-133.

[Forrest et al., 1994] Forrest, S., Perelson, A.S., Allen, L., and R. Cherukuri, S., (1994).Discrimination in a computer, Proceedings of the Symposium on Research in Security and Privacy, 1994, pp. 202–212.

[Förster et al., 2007] Förster, M., Bickel, B., Hardung, B., and Kókai,C. (2007). Self-adaptive ant colony optimization applied to function allocation in vehicle networks_, Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation, 2007, pp. 1991-1998, ACM Press.

[Forsati et al,. 2008] Forsati, R., Mahdavi, M., Kangavari, M., and Safarkhani, B. (2008). Web page clustering using harmony search optimization. In: Canadian conference on electrical and computer engineering, CCECE. pp 1601–1604

[Friedberg,1958] Friedberg, R.M. (1958). A learning machine: Part i. IBM Journal of Research and Development 2:2–13]

[Garey et Johnson, 1979] Garey, M.R., and Johnson, D.D. (1979). Computers and intractability; a guide to the theory of NPcompleteness.W.H. Freeman, 1979.

[Geem et al., 2001] Geem, Z.W., Kim, J.H., Loganathan, G.V. (2001) A new heuristic optimization algorithm: harmony search. J. Simulations 76, 60–68

[Geem et Choi, 2007] Geem, Z., and Choi, JY. (2007). Music composition using harmony search algorithm. In: Giacobini M (ed) Applications of evolutionary computing. Springer, Berlin, pp 593–600

[Geem, 2007] Geem, Z. (2007) Harmony search algorithm for solving sudoku. In: Apolloni B, Howlett RJ, Jain L (eds) Knowledge-based intelligent information and engineering systems, Lecture Notes in Computer Science, vol 4692. Springer, Berlin/Heidelberg, pp 371–378

[Geem et al., 2005a] Geem, ZW., Tseng, CL., and Park, Y. (2005a). Harmony search for generalized orienteering problem: best touring in china. In:Wang L, Chen K, Ong Y (eds) Advances in natural computation. Springer, Berlin, pp 741–750

[Geem et al., 2005b] Geem, ZW., Lee, KS., and Park, Y. (2005b). Application of harmony search to vehicle routing. Am J Appl Sci 2(12):1552–1557

[Glover et Laguna, 1997] Glover, F., and Laguna, M. (1997). Tabu search. Kluwer Academic Publishers.

[Goldberg, 1989] Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning, 1989, Addison-Wesley

[Gomez et Hougen, 2005a] Diaz-Gomez, P.A., Hougen, D.F.(2005). Analysis and mathematical justification of afitness function used in an intrusion detection system, in: H.-G. Beyer, U.-M. O'Reilly (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'05), Washington, DC, USA, 25–29 June 2005, ACM, 2005, pp.1591–1592.

[Gomez et Hougen, 2005b] Diaz-Gomez, P.A.,and Hougen, D.F.(2005),. Analysis of an off-line intrusion detection system:a case study in multi-objective genetic algorithms, in: I. Russell, Z. Markov (Eds.),Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, AAAI Press, Clearwater Beach, FL, USA, 2005, pp.822–823.

[Gomez et Hougen, 2005c] Diaz-Gomez, P.A., and Hougen, D.F.(2005). Improved off-line intrusion detection using agenetic algorithm, in: Proceedings of the Seventh International Conference onEnterprise Information Systems, 2005, pp. 66–73.

[Gomez et Hougen, 2006] Diaz-Gomez, P.A.,and Hougen, D.F.(2006). A genetic algorithm approach for doing misusedetection in audit trail files, in: The 15th International Conference on Computing
(CIC'06), November 2006, IEEE Computer Society, 2006, pp. 329–338.

[Haupt et Haupt, 1998]
Haupt, R.L., and Haupt, S. E. (1998). Practical genetic algorithm. John Wiley & Sons, New York, 1998.

[Heberlein et al., 1994] Heberlein , L.T. , Mukherjee, B., and Levitt, K.N. (1994). "Network Intrusion Detection", IEEE Network Journal, pp. 26-41.

[Hoang, 2008] Hoang, D.T (2008). Metaheuristics for NP-Hard combinatorial optimization Problems. Department of Electrical and Computer Engineering, National University of Singapore.

[Holland, 1975] Holland, J.H. Adaptation in natural and artificial systems. Technical report, University of Michigan, Ann Arbor, 1975.

[Holland, 1992] Holland, J.H. (1992). Adaptation in Natural and Artificial Sytems: An introductory analysis with applications to biology, control, and arti cial intelligence. MIT Press/Bradford books, Cambridge, MA, 1992. 2nd edition.

[Ingber, 1996] Ingber, L .(1996) Adaptive simulated annealing (asa): Lessons learned. Control and Cybernetics 25(1):33–54

[Ingram et Zhang, 2009] Ingram, G., and Zhang, T. (2009). Overview of applications and developments in the harmony search algorithm. In: Geem Z (ed) Music-inspired Harmony search algorithm. Springer Berlin, Heidelberg, pp 15–37

[Johnson et McGeoch, 1997] Johnson, D.SS, and McGeoch, L. A. (1997). The travelling salesman problem: A case study in local optimization. In Local Search in Combinatorial Optimization.Wiley, 1997, pp. 215–310.

[Jourdan , 2003] Laetitia Jourdan (2003). Métaheuristiques pour l'extraction de connaissances: application à la génomique. Thèse de Doctorat.Université des sciences et des techniques de Lille

[Kennedy et Eberhart, 1995] Kennedy, J., and Eberhart, R. C. (1995). Particle swarm optimization. Proceedings of the IEEE Conference on Neural Networks, IV, Piscataway, NJ, pp. 1942-1948.

[Kirkpatrick et al., 1983] Kirkpatrick, S, Gelatt, J.C.D, Vecchi , M. P. (1983). Optimization by simulated annealing. Science 220(4598):671–680.

[Landau, 1909] Landau, E. (1909). Handbuch der Lehre von der Verteilung der

Primzahlen. B. G.Teubner, Leipzig, Germany, reprinted by Chelsea, New York, 1953.

[Larrañaga et Lozano, 2001] Larrañaga,P. Lozano, J.A.. (2001). _Estimation of Distributions Algorithms, a new tool for evolutionary computation, 2001, Kluwer Academic Publishers.

[Lee et al., 2004] Lee, KS., and Geem, ZW. (2004). A new structural optimization method based on the harmony search algorithm.Comput Struct 82(9–10):781–798

[Lourenço et al., 2002] Lourenço, H. R., Martin, O., and Stützle, T. (2002). Iterated local search. In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics. Kluwer Academic Publishers, Norwell, MA, pp.321-353.University of Siegen, Germany.

[Luong, 2011] Luong, T.V. (2011) Métaheuristiques parallèles sur GPU, Thèse de doctorat, Université Lille Nord-de-France- Sciences et Technologies.

[MacArthur et Wilson, 1967] MacArthur, R., and Wilson, E (1967). The Theory of Biogeography. Princeton, NJ: Princeton Univ. Press.

[Majorczyk, 2008] Majorczyk, F. (2008).“Détection d’intrusions omportementale par diversification de COTS : application au cas des serveurs web”. Thèse de doctorat de l’Université de Rennes 1-
N° d’ordre 3827.

[Mé, 1995] Mé, L. (1995). Un Algorithme Génétique pour détecter des Intrusions dans un Système Informatique.VALGO, 95(1); 68-78.

[Mé, 1998] Mé, L (1998). GASSATA, a Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis. In: Proc. of the 1st International Workshop on the Recent Advances in Intrusion Detection (RAID 98). Louvain-la-Neuve, Belgium, pages 14–16.

[Metropolis, 1953] Metropolis, N., Rosenbluth, A., Rosenbluth,M.,Teller, M, and Teller, E. (1953) Equations of state calculations by fast computing machines. Journal of Chemical Physics, 21:1087–1092.

[Michel, 2003] Michel, C. (2003). Langage de description d'attaques pour la détection d'intrusions par corrélation d'événements ou d'alertes en environnement réseau hétérogène, Thèse de doctorat de l’Université de Rennes 1, N° d'ordre: 2943.

[Mitchel, 1998] Mitchel, M. An introduction to genetic algorithms. MIT Press, Cambridge, MA, 1998.

[Mladenovic et Hansen, 1997] Mladenovic, N., and Hansen, P. (1997). Variable neighborhood search. In Computers in Operations Research, 24 (11), pp. 1097-1100.

[Moraga et al., 2006] Moraga, R.J., DePuy and Whitehouse,G.E. (2006). Metaheuristics: A Solution Methodology for Optimization Problems, Handbook of Industrial and Systems Engineering, A.B. Badiru (Ed.).

[Mukherjee et al., 1994] Mukherjee, B., Heberlein, L.T., and Levitt, K.N.(1994). Network intrusion detection. IEEE Network, 8(3) :26_41

[Murata, 2002] Murata (2002). Agent Oriented Self Adaptive Genetic Algorithm, Proceedings of the IASTED Communications and Computer Networks, 2002, pp. 348-353, Acta Press.

[Muttil et Liong, 2004] Muttil, N., and Liong, S.Y. (2004). Superior exploration–exploitation balance in shuffled complex evolution. Journal of Hydraulic Engineering 130(12):1202–1205]

[Nemhauser et Wolsey, 1988] Nemhauser, G, L. and Wolsey , A L. Integer and Combinatorial Optimization. John Wiley & Sons, New York, 1988.

[Osman, 1995] Osman, I.H. (1995). An introduction to metaheuristics. In: LawrenceM,Wilsdon C (eds) Operational Research Tutorial Papers, Stockton Press, Hampshire, UK, pp 92–122, publication of the Operational Research Society, Birmingham, UK.

[Ozyer et al.,2007] Ozyer, T., Alhajj, R., Barker,K. (2007). Intrusion Detection by Integrating Boosting Genetic Fuzzy Classifier and Data Mining Criteria for Rule Pre-Screening. Journal of Network and Computer Applications , pages 99–113.

[Padberg et Rinaldi] Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM, 33:60–100.

[Paenke et al.,2007] Paenke I, Branke J, Jin Y (2007) On the influence of phenotype plasticity on genotype diversity. In: First IEEE Symposium on Foundations of Computational Intelligence (FOCI'07), pp 33–40,

[Papadimitriou et Steiglitz, 1982] Papadimitriou, C. H., and Steiglitz, K. (1982). Combinatorial optimization: Algorithms and complexity. Prentice Hall, New Jersey.

[Pietraszek et Berghe, 1982] Pietraszek, T., and Berghe, V.D. (2005). "Defending Against Injection Attacks through Context-sensitive String Evaluation," In Recent Advances in Intrusion Detection (RAID2005), volume 3858 of Lecture Notes in Computer Science, Seattle, WA, 2005, Springer-Verlag, pp. 124-145.

[Price, 1998] Price, K,. (1998). "Intrusion Detection Pages", Université de Purdue, 1998.

[Puchinger et Raid, 2005] Puchinger, J. and Raid, G. R. (2005) Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. J. Mira and J.R. ´Alvarez (Eds.): IWINAC 2005, LNCS 3562, pp. 41–53, 2005. Springer-Verlag Berlin Heidelberg.

[Rayward-Smith, 1994] Rayward-Smith, V.J. (1994). A unified approach to tabu search, simulated annealing and genetic algorithms. In: Rayward-Smith VJ (ed) Applications of Modern Heuristic Methods – Proceedings of the UNICOM Seminar on Adaptive Computing and Information Processing, Alfred Waller Ltd / Nelson Thornes Ltd / Unicom Seminars Ltd, Henley-on-Thames, UK, vol I, pp 55–78

[Reeves, 1997] Reeves, C.R. (1997). Genetic algorithms for the operations researcher. INFORMS Journal on Computing, 9 :231–250.

[Reynolds, 1987] Reynolds, C.W. (1987). Flocks, herds and schools: a distributed behavioral model, Computer Graphics, Vol. 21, N°4, pp.25-34, 1987

[Robbins et Monro, 1951] Robbins, H., and Monro, S. (1951). A stochastic approximation method. Annals of Mathematical Statistics 22(3):400–407, doi:10.1214/ aoms / 1177729586.

[Scarfone et Mell, 2007] Scarfone, K. and Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). Technical Report SP800-94, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg,U.S. Department of Commerce.

[Schaffer et al., 1990] Schaffer, J.D., Eshelman, L.J., and Offutt, D. (1990). Spurious correlations and premature convergence in genetic algorithms. In: Proceedings of the FirstWorkshop on Foundations of Genetic Algorithms (FOGA), pp 102–112.

[Schnecke et al., 1996] Schnecke, O., and Vornberger. (1996). An Adaptive Parallel Genetic Algorithm for VLSI- Layout Optimization, Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, 1996, pp. 859-868, LNCS, Springer

[Sh et Ws, 2003] SH, O., WS, L (2003). An anomaly Intrusion Detection Method by Clustering Normal User Behavior. Computers & Security, Vol 22, No.7, pages 596-612.

[Shekel, 1971] Shekel, J., (1971) Test functions for multimodal search techniques. In: Proceedings of the Fifth Annual Princeton Conference on Information Science and Systems, Princeton. University Press, pp 354–359.

[Simon, 2008] Simon, D. (2008). Biogeography-Based Optimization. IEEE Trans. on Evol. Comput. 12(6), 712–713

[Simon, 2009] Simon, D. (2009). "A Probabilistic Analysis of a Simplified Biogeography-Based Optimization Algorithm".

[Smith, 2004] Smith, S.S.F (2004.) Using multiple genetic operators to reduce premature convergence in genetic assembly planning. Computers in Industry 54(1):35–49.

[Srivastava et al., 2006] Srivastava, A., Sural, S., and Arun K. Majumdar. (2006).Weighted intratransactionalrule mining for database intrusion detection. In PAKDD, pages 611620.

[Tadjen, 2004] Tjaden, B. C (2004). Fundamentals of Secure Computer Systems. Franklin and Beedle & Associates.

[Taillard et al., 2001] Taillard, E.D., Gambardella, L.M., Gendrau, M., and Potvin, J.Y. (2001). Adaptive memory programming: A unified view of metaheuristics. European Journal of Operational Research 135(1):1–16, doi:10.1016/S0377-2217(00)00268-X].

[Talbi, 2009] Talbi., EG. (2009). Metaheuristics: From design to implementation. Wiley.

[Tombini, 2006] Tombini, E. (2006). Amélioration du Diagnostic en Détection d'intrusions:Etude et Application d'une Combinaison de Méthodes Comportementale et par Scénarios. Thèse de Doctorat de l'Institut National des Sciences Appliquées de Rennes.

[Triki et al., 2005] Triki, E., Collette, Y. and Siarry, P. (2005). A theoretical study on the behavior of simulated annealing leading to a new cooling schedule, European Journal of Operational Research, Vol. 166, N° 1, 2005.

[Ursem, 2003] Ursem, R.K. (2003). Models for evolutionary algorithms and their applications in system identification and control optimization. PhD thesis, Department of Computer Science,University of Aarhus, Denmark.

[Vaessens et al., 1992] Vaessens, R.J.M., Aarts, E.H.L., Lenstra, J.K. (1992). A local search template. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN, pp 67–76.

[Vaessens et al., 1998] Vaessens, R.J.M., Aarts, E.H.L., Lenstra, J.K. (1998). A local search template. Computers and Operations Research 25(11):969–979,doi:10.1016/S0305-0548(97)00093-2.

[Voudouris et Tsang, 1995] Voudouris, C., and Tsang, E. (1995). Guided Local search. Technical Report, CSM-217, Department of Computer Science, University of Essex.

[Wallace, 2005] Wallace, A. (2005) The Geographical Distribution of Animals, vol. 2. Adamant Media Corporation, Boston.

[Wang et al., 2007] Ling Wang, Yin Xu, Yunfei Mao, and Minrui Fei. (2007). Discrete Harmony Search Algorithm. Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronics and Automation, Shanghai University,Shanghai.

[Weise et al., 2008] Weise, T., Zapf,M., Chiong, R. and Nebro, A. J. (2008) Nature-Inspired Algorithms for Optimisation, Series: Studies in Computational Intelligence, Vol. 193, Chiong, Raymond (Ed.),2009, XVIII, 536 p. 161 illus.

[Wood et Erlinger, 2007] Wood, M., and Erlinger, M. (2007). Intrusion detection message exchange requirements. IETF Intrusion Detection Exchange Format Working Group. Request for Comments. Reference : 'rfc4766'.

[Wu et Banzhaf, 2010] Wu, S.X., and Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems : A review. Computer Science Department, Memorial University of Newfoundland, St John's, NL A1B 3X5, Canada. Applied Soft Computing 10 1–35.

[Xu et Zhang, 2005] Xu, B., and Zhang, A (2005). Application of Support Vector Clustering Algorithm to Network Intrusion Detection. In: Proc. of the International Conference on Neural Networks and Brain, ICNN&B '05. Volume 2, pages 1036 – 1040.

[Yang et al., 2011] Haidong, Yang., Jianhua, Guo., Feiqi, Deng (2011). Collaborative RFID Intriusion Detection with an Artificial Immune System. Journal of Intelligent Information System, Volume 36, Number 1, pages 1-26

[Yu, 2002] Yu, T.L. (2002). A survey of estimation of distribution algorithms.Technical report. University of Illinois, USA