UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté de Technologie

Département d'Electronique

MEMOIRE DE MAGISTER

Spécialité : Signaux et Systèmes

IDENTIFICATION DES FILTRES NUMERIQUES PAR UN ALGORITHME DE COLONIE DE FOURMIS

Par

BELLAG KHADIDJA

Devant le jury composé de :

M. BOUNEKHLA	Professeur, USD de Blida	Président
K. BENMANSOUR	Maitre de conférences A, UYF de Médéa	Examinateur
A. GUESSOUM	Professeur, USD de Blida	Rapporteur
M.HADJ SADOK	Maitre de conférences A, USD de Blida	Co-rapporteur
H. BENOUARED	Maitre-assistant A, USD de Blida	Invité

Blida, Novembre 2012

REMERCIEMENTS

Qu'il me soit d'abord permis de remercier et d'exprimer ma gratitude envers le bon dieu, qui m'a donné la force et la patience pour que je puisse terminer ce travail.

Je remercie vivement, Monsieur M'hamed BOUNEKHLA, Professeur au département d'électronique de l'université de Blida, pour m'avoir fait l'honneur de présider mon jury.

Je tiens également à remercier, Monsieur SALHI Hassen, professeur et directeur du laboratoire de recherche S.E.T du département d'électronique de l'université de Blida, ainsi que Monsieur BENMANSOUR Khelifa, maitre de conférence à l'université de Médéa, pour m'avoir fait l'honneur d'être membres de mon jury.

Je tiens ensuite à remercier, Monsieur Abderrezk GUESSOUM, professeur au département d'électronique de l'université de Blida, ainsi que Monsieur M'hammed HADJ SADOK, maitre de conférence au département d'électronique de l'université de Blida, pour avoir dirigé mon travail, et pour leurs nombreux conseils et soutien tout au long de cette thèse.

Je ne me permets pas d'oublier d'exprimer ma très grande reconnaissance envers Monsieur Abdelhalim BENOUARED, chargé de cours au département d'aéronautique de l'université de Blida, pour son aide et ses conseils qui témoignent de ses qualités scientifiques et humaines.

Je remercie également Monsieur Omar BENZINEB, chargé de cours au département d'électronique de l'université de Blida, d'avoir accepté examiner mon travail.

Enfin, je remercie pleinement toute ma famille, pour l'affection, la patience et le soutien inconditionnel, pendant toutes mes années d'étude.

RESUME

La métaheuristique à base d'algorithmes de colonies de fourmis, récemment introduite par A.Colorni, occupe actuellement, une place de plus en plus importante dans le domaine de l'optimisation. Ces algorithmes sont, dans la majorité des travaux rencontrés en littérature, utilisés pour la résolution des problèmes combinatoires. Cependant, en ingénierie, les problèmes d'optimisation sont souvent continus (non dérivabilité de la fonction objectif, multiples minimum locaux, grand nombre de variables…).

Dans ce travail, nous présentons l'algorithme de colonies de fourmis **API**, et son application aux problèmes d'optimisation continue. Cette méthode, basée sur la modélisation du comportement de fourragement d'une espèce de fourmis primitives appelées Pachycondyla Apicalis, opère selon des recherches aléatoires parallèles, à proximité de points appelés sites de chasse. Ces sites sont créés autour d'un point appelé nid. Et à intervalles constants de temps, ce nid est déplacé.

Ceci se traduit par l'établissement de plusieurs recherches aléatoires en parallèle et localisées uniformément dans un sous espace centré en un point (le nid). Ces recherches sont réinitialisées parallèlement, où le point central (nid) est déplacé.

Pour mettre en œuvre l'algorithme **API**, ainsi défini, nous nous sommes proposé dans ce travail, de l'appliquer pour l'identification des filtres numériques.

ملخص

خوارزميات جماعة النمل المقترحة مؤخرا من طرف "كولورني", تشهد اهتماما متزايدا في ميدان التحسين. هذه الخوارزميات مستعملة, في معظم المراجع المتوفرة حاليا, لحل مشاكل التحسين التركيبية. لكن غالبا ما نواجه في ميدان الهندسة مشاكل تحسين مستمرة (نتيجة غير قابلية تفاضل دالة الهدف, كثرة القيم الدنيا المحلية , كثرة المتغيرات...).

في هذا العمل, نقدم خوارزمية جماعة النمل API و تطبيقها لحل مشاكل التحسين المستمرة . هذه الخوارزمية المستلهمة من خلال ملاحظة التصرف الجماعي لفصيلة النمل Pachycondyla Apicalis التي تشتغل وفق أبحاث عشوائية متوازية بالقرب من نقاط تسمى مواقع الصيد والتي تقام حول نقطة تسمى العش . هذا الأخير ينتقل إثر فواصل زمنية ثابتة. يتمثل هذا الأمر في إقامة عدة أبحاث عشوائية متوازية موزعة بانتظام في تحت فضاء ممركز في نقطة, و هي العش. تتعرض هذه الأبحاث بصفة متوازية, لعامل إعادة التشغيل مطابقة مع عامل نقل العش.

بهدف وضع هذه الخوارزمية حيز التطبيق, نقترح في هذا العمل تطبيقها من أجل تحديد المرشحات الرقمية.

Abstract

The metaheuristic based on ant colony algorithms, newly introduced by A.Colorni, occupies nowadays a more and more important place in the field of optimization. These algorithms are, in the majority of the jobs met in literature, used for the resolution of combinatory problems. However, in engineering, problems of optimization are often continuous (no dice rivalry of objective function, local minima, huge number of variables…).

In this work, we introduce the ant colony algorithm, called API, and its application in continuous optimization problems. This method, based on the modelling of the behaviour of fourragement of a species of primitive ants called Pachycondyla Apicalis, works according to researches to unpredictable paralleled researches, close to places called sites of hunting. These sites are created around a point called nest. And at constant intervals of time the nest is moved. This corresponds to a restart approach witch initializes the parallel searches.

In order to implement this algorithm, we use it ,In this work, to identify digital filters.

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION

Les problèmes d'optimisation occupent actuellement une place de plus en plus importante dans les processus décisionnels, et ceci, dans multiples domaines, où le besoin toujours croissant de l'efficacité et de la rentabilité a conduit les décisionnaires à ne plus se fier seulement à l'expérience des experts, mais aussi aux résultats numériques obtenus par la résolution de problèmes d'optimisation.

L'étude basée sur l'observation comportementale des groupes et des individus dans différents domaines, tels que la biologie, l'éthologie, … etc., est une source d'inspiration en ingénierie informatique, où l'étude et la modélisation des systèmes complexes sont très présentes. Cette étude a donné naissance à plusieurs méthodes d'optimisation, appelées « métaheuristiques ».

Nous citons parmi elles, celle basée sur les algorithmes de colonie de fourmis, qui connait actuellement des succès croissants auprès des scientifiques spécialisés en informatique et en recherche opérationnelle [1,2]. Cette métaheuristique connait aussi une multitude d'applications, et ceci dans différents domaines, tels que la robotique, la classification des objets, les réseaux de communication et l'optimisation combinatoire [3,4,5,6,7].

Ces algorithmes sont généralement utilisés comme des méthodes génériques pouvant optimiser une large gamme de problèmes, sans nécessiter de changements profonds dans la procédure employée. Et la majorité d'entre eux s'inspirent des comportements collectifs de dépôt et de suivi de pistes observés dans la colonie de fourmis. Une colonie d'agents simples (les fourmis) communiquent indirectement via des modifications dynamiques de leur environnement (les pistes de phéromone) et construisent ainsi une solution à un problème, en s'appuyant sur leur expérience collective.

Le premier algorithme de colonies de fourmis apparu, est le ″Ant System″ (AS) [2], implémenté pour résoudre le problème du voyageur de commerce (Travelling Sailesman), qui est en fait un problème d'optimisation combinatoire. Suite à cet algorithme, pleins d'autres sont apparus [8,9,10,12,13] traitant toujours des

problèmes d'optimisation combinatoire. Cependant, il existe en ingénierie, une large gamme de problèmes où la fonction objectif est à variables continues, et pour lesquels, la métaheuristique peut être d'un grand secours (fonction non dérivable, multiples minimums locaux, non convexité, nombre de variables important,..).

De ce fait, plusieurs tentatives d'adaptation des algorithmes de colonies de fourmis aux cas continues sont apparues [14, 15, 16, 17].

Dans ce mémoire, nous nous intéressons à appliquer un des algorithmes de colonies de fourmis à l'identification des filtres numériques, qui est l'algorithme API, déduit de la modélisation du comportement de fourragement de l'espèce *Pachycondyla Apicalis.*

Comme pour tous les problèmes d'identification ou d'optimisation, une connaissance assez riche du processus à optimiser est nécessaire. Nous avons consacré le **premier chapitre** de ce mémoire à l'étude des filtres numériques. Nous y présenterons ainsi, d'une manière succincte, des rappels qui seront utilisés tout au long de ce travail. Nous y présenterons aussi les caractéristiques des filtres numériques et une classification des différentes méthodes de leur synthèse.

Le **chapitre deux** décrit d'abord le cadre des ″métaheuristiques″ et leur apport aux problèmes d'optimisation difficile. Puis présente un état de l'art sur les métaheuristiques dites « de colonies de fourmis ».

Le chapitre trois est consacré à l'étude de l'algorithme de colonies de fourmis *Pachycondyla Apicalis* **API,** et son application au problème général d'optimisation. L'atout de cet algorithme, par rapport aux autres, est le fait qu'il soit adaptable aux cas continus et discrets, et qu'il ne fait pas usage de la communication indirecte par pistes de phéromone.

Dans le **chapitre quatre**, nous mettons en œuvre l'algorithme de colonies de fourmis **API** pour l'identification des filtres numériques. Ce chapitre comportera une description des caractéristiques des filtres numériques à identifier, une étude sur la fonction objectif appropriée, une étude détaillée sur l'adaptation, étape par étape, de l'algorithme au problème d'optimisation considéré. Nous présenterons, par la suite,

les résultats obtenus munis de discussions. Enfin, nous étudierons l'influence de certains paramètres de l'algorithme **API** sur la performance de sa convergence vers l'optimum global.

CHAPITRE 1

ALGORITHMES DE COLONIES DE FOURMIS EN OPTIMISATION

ETAT DE L'ART

1-1 .Introduction

Les problèmes d'optimisation occupent actuellement une place importante dans la communauté scientifique. Ces problèmes peuvent être combinatoires (discrets) ou à variables continues, avec un seul ou plusieurs objectifs, statiques ou dynamiques.

Un problème d'optimisation est défini par un ensemble de variables, une fonction objectif et un ensemble de contraintes [2].

L'espace de recherche, qui est l'ensemble des domaines de définition des différentes variables du problème.

La fonction objectif définit le but à atteindre. On cherche à minimiser ou maximiser celle-ci.

L'ensemble des contraintes est en général un ensemble d'égalités ou d'inégalités que les variables de l'espace d'état doivent satisfaire.

Les méthodes d'optimisation recherchent un point ou un ensemble de points dans l'espace de recherche qui satisfait l'ensemble des contraintes et qui minimisent ou maximisent la fonction objectif.

La méthode d'optimisation avec laquelle nous avons travaillé tout au long de notre mémoire est une métaheuristique dite « colonies de fourmis ».

Ce chapitre donne un bref aperçu sur la famille des métaheuristiques, puis présente un état de l'art sur les colonies de fourmis.

1-2- Métaheuristiques.

Les métaheuristiques sont une famille d'algorithmes stochastiques destinés à résoudre des problèmes d'*optimisation difficile.*

Utilisés dans de nombreux domaines, ces méthodes présentent l'avantage d'être généralement efficaces sans pour autant que l'utilisateur ait à modifier la structure de base de l'algorithme qu'il utilise [1].

Ces méthodes ont en commun, les caractéristiques suivantes [2,3] :

- Elles sont, au moins pour une partie, stochastiques : ce qui permet de faire face au problème de l'explosion combinatoire des possibilités.

- Généralement d'origine discrète, elles ont l'avantage décisif dans le cas continu d'être discrètes, c'est-à-dire qu'elles ne recourent pas au calcul souvent problématique des gradients de la fonction objectif.

- Elles sont inspirées par des analogies : avec la physique (recuit simulé), avec la biologie (algorithmes évolutionnaires, recherche avec tabou,..) ou avec l'éthologie (algorithmes de colonies de fourmis, essaims particulaires…).

- Elles surmontent le piège des minimums locaux en autorisant de temps en temps des mouvements de montées, autrement dit, en acceptant une dégradation temporaire de la fonction objectif au cours de leur progression. Un mécanisme de contrôle des dégradations spécifique à chaque métaheuristique, permet d'éviter la divergence du procédé.

Pour compléter cette présentation succincte des métaheuristiques, on peut souligner une autre richesse : elles se prêtent à toutes sortes d'extensions. Citons en particulier [2,3] :

- L'optimisation multi-objectifs [4], où il s'agit d'optimiser simultanément plusieurs objectifs contradictoires. Il n'existe pas dans ce cas, un optimum unique, on cherche en revanche une gamme de solutions « optimales au sens de Pareto » qui forment «la surface de compromis » du problème.

- L’optimisation multimodale, où l’on s’efforce de repérer tout un jeu d’optimums globaux ou locaux. Les algorithmes évolutionnaires sont particulièrement bien adaptés à cette tâche, de par leur nature distribuée.

-L’optimisation dynamique, qui fait face à des variations temporelles de la fonction objectif au cours du processus d’optimisation.

-La parallélisation, où la charge de calcul de l’optimisation est répartie sur des unités fonctionnant en concert afin d’accélérer l’optimisation.

- L’hybridation, qui vise à combiner des métaheuristiques dans le but de répartir les tâches de recherche au profit des avantages de chaque métaheuristique [5].

Enfin, les métaheuristiques sont souvent employées pour leur facilité de programmation et de manipulation. Elles sont le plus judicieusement employées sur des problèmes d’optimisation difficile, où des méthodes d’optimisation classiques montrent leurs limites.

De façon générale, nous pouvons dire que des problèmes d’optimisation présentant les caractéristiques suivantes sont assez propices à l’utilisation des métaheuristiques [2].

- Complexité élevée.
- Nombreux optima locaux.
- Discontinuités.
- Fortes contraintes.
- Non dérivabilité.
- Temps de calcul de la fonction objectif prohibitif.
- Solution approchée souhaitée.

Dans ce qui suit, nous allons décrire la métaheuristique dite colonie de fourmis.

1-3- Algorithmes de colonies de fourmis.

Les algorithmes de colonies de fourmis forment une classe des métaheuristiques proposée pour l’optimisation difficile [11], s’inspirant des comportements des fourmis observés dans la nature ; notamment l’utilisation de la communication indirecte (à travers l’environnement), les traces de phéromones que les fourmis laissent dans

leur passage et la construction itérative d'une solution globale qui se base sur une sorte d'intelligence collective.

1-3-1- Relation avec l'informatique

En observant une colonie de fourmis à la recherche de la nourriture aux environs de son nid, on s'aperçoit qu'elle résout des problèmes tels que celui de la recherche du plus court chemin.

Les fourmis résolvent des problèmes complexes par des mécanismes assez simples à modéliser. Il est ainsi assez simple de simuler leur comportement par des algorithmes [6].

1-3-2- Caractéristiques des fourmis.

Les biologistes ont étudié comment les fourmis arrivent à résoudre collectivement des problèmes trop complexes pour un seul individu, notamment les problèmes de choix lors de l'exploitation des sources de nourriture [1].

Les fourmis ont la particularité d'employer pour communiquer des substances volatiles appelées phéromones, elles sont très sensibles à ces substances qu'elles perçoivent grâce à des récepteurs situés dans leurs antennes. Ces substances sont nombreuses et varient selon les espèces.

Les fourmis peuvent déposer des phéromones au sol, grâce à une glande située dans leur abdomen et forment ainsi des pistes odorantes qui pourront être suivies par leurs congénères.

Figure1.1 : Des fourmis suivant une piste de phéromone.

(Photographie de Taillard. E, tirée de [7]).

Les fourmis utilisent les pistes de phéromones pour marquer leur trajet, par exemple entre le nid et une source de nourriture.

Une colonie est ainsi capable de choisir le plus court chemin vers une source de nourriture à exploiter [8,9], sans que les individus aient une vision globale du trajet.

Ce concept a été démontré par l'expérience du *pont à double branches* qui a été réalisée par *Deneubourg et al*. [11]. Il s'agit d'un pont à deux branches de longueurs différentes placé entre le nid et une source de nourriture.

Au début de l'expérience, les fourmis choisissent une des deux branches avec des probabilités identiques, car la piste ne contient pas de phéromone (figure (1.2. (a)).

Les fourmis qui empruntent la branche la plus courte sont les plus rapides à arriver au nid, après avoir visité la source de nourriture. En conséquence, la quantité de phéromone présente sur le plus court trajet est légèrement plus importante que celle présente sur le trajet le plus long. Donc, le trajet le plus court a une probabilité plus grande d'être emprunté (Figure (1.2 (b)). Après un certain temps, toutes les fourmis choisissent le plus court trajet entre le nid et la source de nourriture (Figure (1.2 (c)).

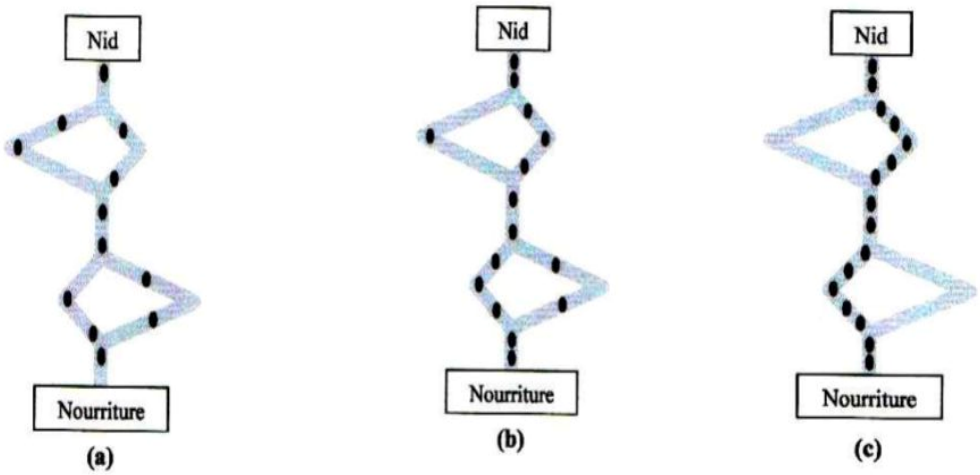


Figure 1.2 : L'expérience du pont à double branches.

Une autre expérience dite l'*expérience de l'obstacle* a été réalisée. Elle montre que les fourmis sont capables d'effectuer des détournements si un obstacle a été placé sur leur chemin entre la source de nourriture et le nid, pour retrouver le plus court chemin [12].

Lorsqu'un obstacle est posé sur le chemin des fourmis, elles choisissent d'aller à gauche ou à droite avec des probabilités égales (Figure 1.3(b)). Le chemin le plus court sera le plus rapidement parcouru par les fourmis (Figure 1.3(c)). En conséquence, la plus grande quantité de phéromone est posée rapidement sur le plus court chemin, et puisque les fourmis choisissent le trajet qui contient le plus de phéromone, le plus court chemin sera utilisé.

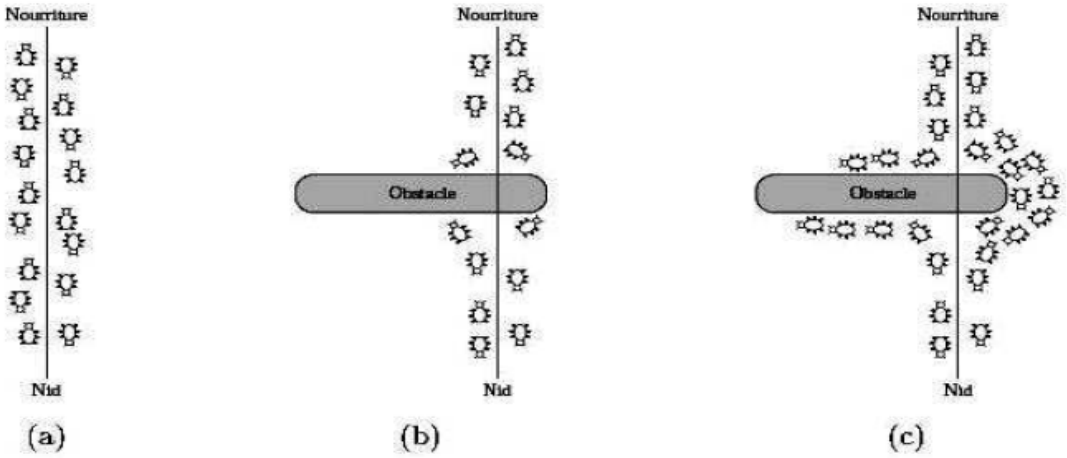


Figure 1.3 : L'expérience de l'obstacle.

Il est difficile de connaitre avec précision les propriétés physico-chimiques des pistes de phéromones qui varient en fonction des espèces et d'un grand nombre de paramètres. Cependant, les métaheuristiques d'optimisation de colonies de fourmis s'appuient en grande partie sur le phénomène d'évaporation des pistes de phéromones, en tenant compte que dans la nature, les pistes s'évaporent plus lentement que ne le prévoient les modèles.

Les fourmis réelles disposent, en effet, d'heuristiques leur apportant un peu plus d'informations sur le problème. Il faut garder à l'esprit que l'intérêt immédiat de la colonie (trouver le plus court chemin vers une source de nourriture) peut être en concurrence avec l'intérêt adaptatif de tels comportements.

1-3-3- Fourmis réelles et fourmis virtuelles.

Dans ce paragraphe, nous allons exposer les différences et les points communs entre les fourmis réelles et les fourmis virtuelles [2,6], ceci expliquera comment les fourmis virtuelles peuvent être exploitées pour résoudre des problèmes d'optimisation.

Les fourmis virtuelles ont une double nature. D'une part, elles modélisent le comportement abstrait des fourmis réelles, et d'une autre part, elles peuvent être

enrichies par des capacités que ne possèdent pas les fourmis réelles, afin de les rendre plus efficaces.

Points communs.

- Colonie d'individus coopérants :

Comme pour les fourmis réelles, une colonie virtuelle est un ensemble d'entités non-synchronisées qui se rassemblent ensemble pour trouver une "bonne" solution au problème considéré. Chaque groupe d'individus doit pouvoir trouver une solution même si elle est mauvaise.

- Pistes de phéromones :

Ces entités communiquent par le mécanisme des pistes de phéromone. Cette forme de communication joue un grand rôle dans le comportement des fourmis : son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis, en fonction de l'historique laissé par ces phéromones.

- Evaporation des phéromones :

Ce mécanisme permet d'oublier lentement ce qui s'est passé avant, c'est ainsi que la fourmi peut diriger sa recherche vers de nouvelles directions, sans être trop contrainte par ses anciennes décisions.

- Recherche du plus petit chemin :

Les fourmis réelles et virtuelles partagent un but commun: recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).

- Déplacements locaux :

Les vraies fourmis ne sautent pas de cases, tout comme les fourmis virtuelles. Elles se contentent de se déplacer entre des sites adjacents du terrain.

- Choix aléatoire lors des transitions :

Lorsqu'elles sont sur un site, les fourmis réelles et virtuelles doivent décider sur quel site adjacent se déplacer. Cette prise de décision se fait au hasard et dépend de l'information locale déposée sur le site courant. Elle doit tenir compte des pistes de phéromones, mais aussi du contexte de départ (ce qui revient à prendre en considération les données du problème d'optimisation combinatoire pour une fourmi virtuelle).

Différences.

- Elles vivent dans un monde non-continu :

Leurs déplacements consistent en des transitions d'état.

- Mémoire (état interne) de la fourmi :

Les fourmis réelles ont une mémoire très limitée, tandis que nos fourmis virtuelles mémorisent l'historique de leurs actions. Elles peuvent aussi retenir des données supplémentaires sur leurs performances.

- Nature des phéromones déposées :

Les fourmis réelles déposent une information physique sur la piste qu'elles parcourent, là où les fourmis virtuelles modifient des informations dans les variables d'états associées au problème. Ainsi, l'évaporation des phéromones est une simple décrémentation de la valeur des variables d'états à chaque itération.

- Qualité de la solution :

Les fourmis virtuelles déposent une quantité de phéromones proportionnelle à la qualité de la solution qu'elles découvrent.

- Retard dans le dépôt de phéromone :

Les fourmis virtuelles peuvent mettre à jour les pistes de phéromones de façon non immédiate : souvent elles attendent d'avoir terminé la construction de leur solution. Ce choix dépend du problème considéré bien évidemment.

- Capacités supplémentaires :

Les fourmis virtuelles peuvent être pourvues de capacités artificielles afin d'améliorer les performances du système. Ces possibilités sont liées au problème et peuvent être :

1. l'anticipation : pour faire son choix, la fourmi étudie les états suivants et non seulement l'état local.

2. le retour en arrière : une fourmi peut revenir à un état déjà parcouru si la décision prise à cet état a été mauvaise.

1-4- Algorithme de colonie de fourmis : formalisation et propriétés

> *Une métaheuristique de colonie de fourmis est un processus stochastique construisant une solution, en ajoutant des composants aux solutions partielles.Ce processus prend un compte (i) une heuristique sur l'instance du problème (ii) des pistes de phéromone changeant dynamiquement pour refléter l'expérience acquise par les agents.*

Cet extrait traduit du grand livre « *handbook of metaheuristics* » [1], est une description élégante s'appliquant aux problèmes (combinatoires) où une construction partielle de la solution est possible.

Une formalisation plus précise y existe. Elle passe par une représentation du problème, un comportement de base des fourmis et une organisation générale de la métaheuristique.

Plusieurs concepts sont également à mettre en valeur pour comprendre les principes de ces algorithmes, notamment la définition des pistes de phéromones en tant que mémoire adaptative, la nécessité d'un réglage intensification/diversification et enfin l'utilisation d'une recherche locale. Nous traitons ci-après ces différents sujets.

1-4-1- Formalisation

Représentation du problème

Le problème est représenté par un jeu de solutions, une fonction objectif assignant une valeur à chaque solution et un jeu de contraintes.

L'objectif est de trouver l'optimum global de la fonction objectif satisfaisant les contraintes. Les différents états du problème sont caractérisés comme une séquence de composants. On peut noter que, dans certains cas, un coût peut être associé à des états autres que des solutions. Dans cette représentation, les fourmis construisent des solutions en se déplaçant sur un graphe G=(C ;L), où les nœuds sont les composants de C et où l'ensemble L connecte les composants de C. Les contraintes du problème sont implémentées directement dans les règles de déplacement des fourmis (soit en empêchant les mouvements qui violent les contraintes, soit en pénalisant de telles solutions).

Comportement des fourmis.

Les fourmis artificielles peuvent être caractérisées comme une procédure de construction stochastique construisant des solutions sur le graphe G tel que G=(C ; L). En général, les fourmis tentent d'élaborer des solutions faisables, mais, si nécessaire, elles peuvent produire des solutions infaisables.

Les composants et les connexions peuvent être associés à des pistes de phéromones (mettant en place une mémoire adaptative décrivant l'état du système) et à une valeur heuristique (représentant une information à priori sur le problème, ou venant d'une source autre que celle des fourmis). Les pistes de phéromone et la valeur de l'heuristique peuvent être associées soit aux composants, soit aux connexions.

Chaque fourmi dispose d'une mémoire utilisée pour stocker le trajet effectué, d'un état initial et de conditions d'arrêt. Les fourmis se déplacent d'après une règle de décision probabiliste fonction des pistes de phéromone locales, de l'état de la fourmi et des contraintes du problème. Lors de l'ajout d'un composant à la solution en cours, les fourmis peuvent mettre à jour la piste de phéromone des composants ou

des connexions utilisées. Enfin, une fourmi dispose au minimum de la capacité de construire une solution au problème.

Organisation de la métaheuristique

En plus des règles régissant le comportement des fourmis, un autre processus majeur a cours : l'évaporation des pistes de phéromone. En effet, à chaque itération, la valeur des pistes de phéromone est diminuée. Le but de cette diminution est d'éviter une convergence trop rapide et le piégeage de l'algorithme dans des minimums locaux, par une fourmi d'oubli favorisant l'exploration de nouvelles régions.

1-4-2- Phéromone et mémoire.

Les pistes de phéromone décrivent à chaque pas l'état de la recherche de la solution par le système, les agents modifient la façon dont le problème va être représenté et perçu par les autres agents. Cette information est partagée par le biais des modifications de l'environnement des fourmis, grâce à une forme de communication indirecte : la strigmergie. L'information est donc stockée un certain temps dans le système, ce qui a amené certains auteurs à considérer ce processus comme une forme de mémoire adaptative [10], où la dynamique de stockage et de partage de l'information va être cruciale pour le système.

Le choix de la méthode d'implémentation des pistes de phéromone est donc important pour obtenir les meilleurs résultats. Ce choix est en grande partie lié aux possibilités de représentation de l'espace de recherche, chaque représentation pouvant apporter une façon différente d'implémenter les pistes.

1-4-3- Intensification/diversification.

Lors de la conception et l'utilisation d'un algorithme de colonie de fourmis, on est souvent confronté au problème de l'emploi relatif des processus d'intensification et de diversification. Par *intensification*, on entend l'exploitation de l'information accumulée par le système à un moment donné. La *diversification* est au contraire l'exploration de régions de l'espace de recherche imparfaitement prises en compte.

Bien souvent, il s'agit de choisir quand et où injecter de l'aléatoire dans le système *(diversification)* et/ou améliorer une solution *(intensification).*

Il existe plusieurs façons de gérer l'emploi de ces deux facettes des métaheuristiques d'optimisation. La plus évidente passe par le réglage via les deux paramètres α et β, qui déterminent relativement l'influence relative des pistes de phéromone et de l'information heuristique. Plus la valeur de α sera élevée, plus l'intensification sera importante, car plus les pistes auront une influence sur le choix des fourmis. A l'inverse, plus α sera faible, plus la diversification sera forte, car les fourmis éviteront les pistes. Le paramètre β agit de façon similaire. On doit donc gérer conjointement les deux paramètres, pour régler ces aspects.

Le choix *Intensification/diversification* peut aussi s'effectuer de manière statique avant le lancement de l'algorithme, en utilisant une connaissance à priori du problème, ou de manière dynamique, en laissant le système décider du meilleur réglage.

1-4-4- Recherche locale et heuristique.

Les métaheuristiques de colonies de fourmis sont souvent plus efficaces quand elles sont hybridées avec des algorithmes de recherche locale. Ceux-ci optimisent les solutions trouvées par les fourmis, avant que celles-ci ne soient utilisées pour la mise à jour des pistes de phéromone. Du point de vue de la recherche locale, utiliser des algorithmes de colonies de fourmis pour engendrer une solution initiale est un avantage indéniable.

Une autre possibilité pour améliorer les performances est d'injecter une information heuristique plus pertinente. Cet ajout a généralement un coût élevé en termes de calculs supplémentaires.

1-4-5- Parallélisme.

La structure même des métaheuristiques de colonies de fourmis comporte un parallélisme intrinsèque. D'une manière générale, les solutions de bonne qualité émergent du résultat des interactions indirectes ayant cours dans le système, et non pas d'un codage explicite d'échanges. En effet, chaque fourmi ne prend en compte que des informations locales de son environnement (les pistes de phéromone) ; il est

donc facile de paralléliser un tel algorithme. Il est intéressant de noter que les différents processus en cours dans la métaheuristique (i.e. le comportement des fourmis, l'évaporation et les processus annexes) peuvent également être implémentés de manière indépendante, l'utilisateur étant libre de décider de la manière dont ils vont interagir.

1-4-6- Convergence.

Les métaheuristiques peuvent être vues comme des modifications d'un algorithme de base : une recherche aléatoire. Cet algorithme possède l'intéressante propriété de garantir que la solution optimale sera trouvée tôt ou tard, on parle alors de convergence. Cependant, puisque cet algorithme de base est biaisé, la garantie de convergence n'existe plus.

Si, dans certains cas, on peut facilement être certain de la convergence d'un algorithme de colonies de fourmis (MMAS par exemple), le problème reste entier en ce qui concerne la convergence d'un algorithme ACO quelconque. Cependant, il existe une variante dont la convergence a été prouvée [13,14] : ''le graph-based Ant system''(GBAS). La différence entre (GBAS) et l'algorithme AS se situe au niveau de la mise à jour des pistes de phéromone, qui n'est permise que si une meilleure solution est trouvée. Pour certaines valeurs de paramètres, et étant donné $\varepsilon>0$ une ''faible'' valeur, l'algorithme trouvera la solution optimale avec une probabilité $P_t \geq 1-\varepsilon$, après un temps $t \geq t_0$, (où t_0 est fonction de ε).

1-5- Problème du voyageur de commerce et le Ant System.

Le problème du voyageur de commerce (PVC) ou *Travelling Salesman Problem (TSP),* est un classique du genre. Ce problème a fait l'objet de la première implémentation d'un algorithme de colonies de fourmis : le ''Ant System'' (AS) [15].

Rappelons tout de même sa formulation générale :

Définition:

Un voyageur de commerce doit visiter un ensemble $\{ V_1,\ldots..,V_n \}$ de n villes dont on connait les distances respectives $d(V_i, V_j)$, $\forall (i,j) \in \{1,\ldots..,n\}$. Le problème consiste à trouver la permutation σ telle que la séquence $s = (V_{\sigma(1)}, \ldots.., V_{\sigma(n)})$ minimise la distance totale D(σ) parcourue par le voyageur.

$$D(\sigma) = \sum_{i=1}^{n-1} d(V_{\sigma(i)}, V_{\sigma(i+1)}) + d(V_{\sigma(n)}, V_{\sigma(1)}) \qquad (1.1)$$

L'espace de recherche est l'ensemble des combinaisons possibles des *n* villes, soit au total $n!$ combinaisons. Ce problème N-P difficile [20], peut être aussi considéré comme la recherche d'un circuit hamiltonien de longueur minimale dans un graphe complet pouvant être antisymétrique dans le cas général $\exists (i,j)$ tel que $d(V_i,V_j) \neq d(V_j,V_i)$.

1-6- Autres problèmes combinatoires.

Les algorithmes de colonies de fourmis sont très étudiés depuis quelques années et ceci dans un grand nombre d'applications. Ainsi, plusieurs variantes ont été produites.

Dans les deux principaux champs d'application (problèmes NP-difficiles et problèmes dynamiques), certains algorithmes ont donné de très bons résultats. On peut notamment retenir des performances particulièrement intéressantes dans le cas de l'affectation quadratique [16,17,18] de problèmes de planification [1], de coloriage de graphes [19], d'affectation de fréquence [20], ou du routage sur réseau [8].

1-7- Adaptation aux cas continu.

Les métaheuristiques de manière générale, et les algorithmes de colonies de fourmis particulièrement sont bien souvent élaborés pour des problèmes combinatoires, mais il existe une classe de problèmes souvent rencontrée en ingénierie, où la fonction objectif est à variables continues, et pour lesquels les métaheuristiques peuvent être d'un grand secours (fonction non dérivable, multiples minimums locaux, grand nombre de variables, non convexité, etc.). Plusieurs

tentatives pour adapter les métaheuristiques de colonies de fourmis au domaine continu sont apparues [21, 22, 23, 24].

Nous avons recensé dans la littérature plusieurs algorithmes de colonies de fourmis pour l'optimisation continue, on citera entre autres :

1-7-1- L'algorithme CACO

Le premier des algorithmes de colonies de fourmis pour les cas continus, nommé tout naturellement ''Continuous Ant Colony System'' CACO [21,25], utilise deux approches : un algorithme de type évolutionnaire sélectionne et croise des régions d'intérêt, que des fourmis explorent et évaluent. Une fourmi sélectionne une région avec une probabilité proportionnelle à la concentration en phéromone, de la même manière que dans le « Ant System », une fourmi sélectionnerait une piste allant d'une ville à une autre :

$$p_i\,(t) = \frac{T_i^{\alpha}(t).\ \eta_i^{\beta}\ (t)}{\sum_{j=1}^{N} T_j^{\alpha}\,(t).\ \eta_j^{\beta}(t)} \qquad (1.2)$$

Où N est le nombre de régions et η_i^{β} est utilisée pour inclure une heuristique spécifique au problème. Les fourmis partent alors du centre de la région et se déplacent selon une direction choisie aléatoirement, tant qu'une amélioration de la fonction objectif est trouvée.

Le pas de déplacement utilisé par la fourmi entre chaque évaluation est donné par

$$\delta r(t,R) = R.\left(1 + u^{\left(1-\frac{t}{T}\right)^{c}}\right). \qquad (1.3)$$

Où R est le diamètre de la région explorée, $u \in [0,1]$ un nombre aléatoire, T le nombre total d'itérations de l'algorithme et c un paramètre de refroidissement (permettant de réduire le pas à chaque itération). Si la fourmi a trouvé une meilleure solution, la région est déplacée de façon à ce que son centre coïncide avec cette solution, et la fourmi augmente la quantité de phéromone de la région proportionnellement à l'amélioration trouvée. L'évaporation des « pistes » se fait classiquement en fonction d'un coefficient ρ.

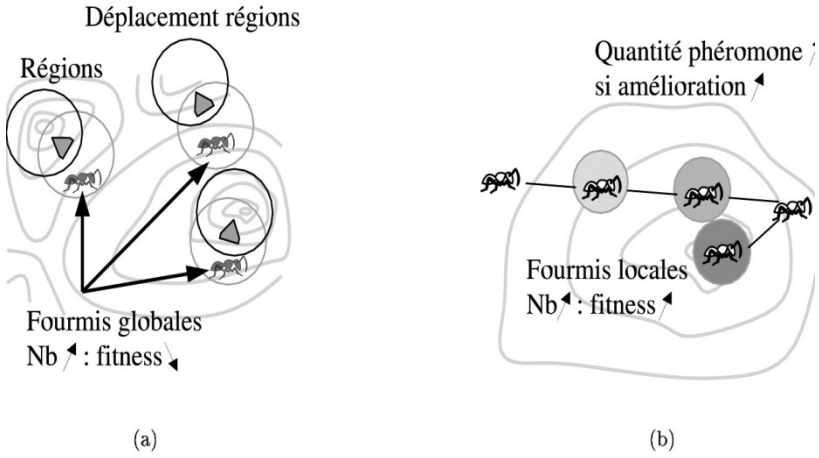


Figure 1.4 – L'algorithme CACO : les fourmis globales (a) participent au déplacement des régions que les fourmis locales (b) évaluent.

1-7-2- Une méthode hybride

Une méthode hybride non baptisée, utilisant à la fois une approche de colonies de fourmis et un algorithme évolutionnaire, a été proposée par Ling et Al. [26]. Mais peu de résultats sont disponibles jusqu'à ce jour. L'idée principale de cette méthode est de considérer les écarts entre deux individus sur chaque dimension comme autant de parties d'un chemin où les phéromones sont déposées, l'évolution des individus étant prise en charge par des opérateurs de *mutation* et de *croisement.*

D'un certain point de vue, cette méthode tente donc de reproduire le mécanisme de construction de la solution composant par composant.

1-7-3- L'algorithme ACO pour l'optimisation continue

Cet algorithme [10] tente de maintenir la construction itérative des solutions dans le cas de variables continues, en adoptant un point de vue différent des précédents.

En effet, il prend le parti de considérer que les composants de toutes les solutions sont formés par les différentes variables optimisées. De plus, plutôt que de

considérer l'algorithme du point de vue de la fourmi, il prend le parti de se placer au niveau de la colonie, les fourmis n'étant plus que des points à évaluer.

Dans cette méthode, dénommée simplement '' ACO pour l'optimisation continue'', on tire aléatoirement une population de fourmis dans une distribution de probabilité à chaque itération. De cet ensemble de points, ne sont conservés que les meilleurs points, qui servent alors à construire une meilleure distribution de probabilités. La distribution de probabilités ici utilisée est un '' amalgame pondéré de noyaux normaux''.

1-7-4- L'algorithme CACS

Cet algorithme appelé '' continuous ant colony system'' est très proche du précédent. En effet, dans CACS comme dans '' ACO pour l'optimisation continue'', le cœur de l'algorithme consiste à faire évoluer une distribution de probabilités. La distribution ici utilisée est dite ''normale''.

1-7-5- L'algorithme CIAC

Un algorithme, se focalisant sur les principes de communication des colonies de fourmis a été proposé pour l'optimisation des fonctions continues dans [27,28].

Cet algorithme appelé CIAC pour '' Continuous Interacting Ant Colony'', inspiré du concept d'hétérarchie dense introduite par Wilson en 1988 [29], utilise deux canaux de communication observables dans la nature :

- Le canal stigmergique classique : qui fait appel à des spots de phéromone, déposés sur l'espace de recherche, qui vont être plus au moins attractifs pour les fourmis, selon leurs concentrations et leurs distances.

- Le canal direct : qui est implémenté sous la forme d'échange de messages entre deux individus. Une fourmi possède une pile de messages reçus et peut en envoyer à une autre fourmi.

1-7-6- L'algorithme API

Dans tous les algorithmes évoqués jusqu'ici, le terme ''colonie de fourmis'' s'entend de par l'utilisation de la stigmergie comme processus d'échange d'informations. Il existe cependant un algorithme adapté au cas continu qui s'inspire du comportement de fourmis primitives de l'espèce *Pachycondyla Apicalis* (APIcalis), et qui ne fait pas usage de la communication indirecte par pistes de phéromone.

Dans cette méthode, on commence par positionner un nid aléatoirement sur l'espace de recherche, puis des fourmis y sont distribuées aléatoirement. Ces fourmis vont alors explorer localement leur '' site de chasse'' en évaluant plusieurs points dans un périmètre donné (figure 1.5). Chaque fourmi mémorise le meilleur point trouvé. Si lors de l'exploration de son site de chasse, elle trouve un meilleur point, alors elle reviendra sur ce site, sinon, après un certain nombre d'explorations, elle choisira un autre site. Une fois les explorations des sites de chasses terminées, des fourmis tirées au hasard comparent deux à deux (comme peuvent le faire les fourmis réelles) leurs meilleurs résultats, puis mémorisent le meilleur des deux sites de chasse. Le nid est finalement réinitialisé sur le meilleur point trouvé après un temps donné, la mémoire des sites des fourmis est remise à zéro, et l'algorithme effectue une nouvelle itération.

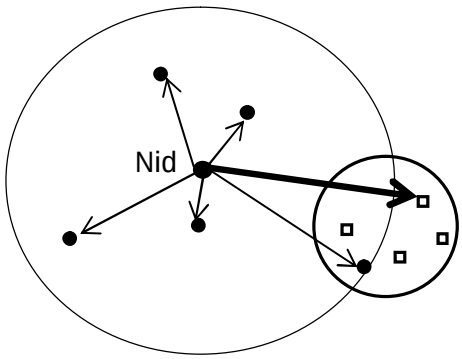


Figure 1.5 – L'algorithme API, une méthode à démarrage multiple inspirée par une espèce de fourmi primitive. Les fourmis (cercles pleins) explorent des sites de chasse (petits carrés) dans un périmètre (grand cercle) autour du nid. Le nid est déplacé sur le meilleur point au moment de la réinitialisation (flèche en trait gras).

1-8- Conclusion

Les métaheuristiques s'inspirant des colonies de fourmis commencent à être bien décrites et formalisées, elles ont été appliquées avec succès à de nombreux problèmes combinatoires et commencent à être adaptées à des problèmes continus.

La recherche étant encore à ses débuts. Les algorithmes produits n'ont pas atteint leur pleine maturité et ne sont donc pas encore compétitifs par rapport à d'autres classes de métaheuristiques plus élaborées sur les systèmes continus.

La suite de ce travail sera consacrée à l'étude de l'algorithme de colonies de fourmis API brièvement décrit dans le paragraphe précédent, et à son application au problème d'optimisation « difficile » à variables continues.

CHAPITRE 2

L'ALGORITHME DE COLONIES DE FOURMIS API

2-1- Introduction

Ce chapitre est entièrement consacré à l'étude d'un des algorithmes de colonies de fourmis brièvement décrits dans le chapitre précédent, il s'agit de l'algorithme API s'inspirant du comportement de fourmis primitives de l'espèce *Pachycondyla Apicalis,* avec lequel nous avons procédé à l'identification des filtres numériques. Nous présenterons ainsi, une modélisation du comportement d'une population de cette espèce et son application au problème général d'optimisation. Cette étude trouve son origine dans les travaux de Dominique Fresneau sur la stratégie de fourragement originale de cette espèce de fourmis ponérine [30].

L'intérêt de ces fourmis pour l'optimisation vient du fait qu'elles utilisent des principes relativement simples à la fois d'un point de vue global et local pour rechercher leurs proies. A partir de leurs nids, elles couvrent globalement une surface donnée en la partitionnant en plusieurs sites de chasse individuels. Pour une fourmi donnée, on observe une stratégie d'exploration aléatoire des sites, sensible au succès rencontré.

Ces principes peuvent être repris pour résoudre un problème analogique qui est la recherche d'un minimum global, par exemple d'une fonction f définie de R dans R.

2-2- Biologie de *Pachycondyla Apicalis*

Cette section donne un aperçu de la biologie de *Pachycondyla Apicalis* et notamment de leur stratégie de recherche de nourriture (le fourragement). Une étude très complète leur est consacrée par Dominique Fresneau de l'université Paris XIII [30].

Pachycondyla Apicalis est une ponérine néotropicale que l'on rencontre en Amérique du Sud, en particulier au Mexique. Sa morphologie et le peu de communication entre individus la classent parmi les fourmis dites primitives, mais certains aspects de son

adaptation démentent ce jugement. Les fourmis *Pachycondyla Apicalis* vivent en petites colonies comportant quelques dizaines d'individus (40 à 100 ouvrières).

Le nid est installé dans de vieilles souches ou dans des arbres morts en décomposition, ce qui correspond à un habitat instable pour des fourmis qui ne savent pas construire de fourmilière. Quand la vétusté de leur nid devient trop importante, elles doivent déménager et chercher un nouveau refuge.

Leur nourriture se compose de petits insectes ou de cadavres d'insectes. Les ouvrières prospectent individuellement autour de la fourmilière et ramènent les proies au nid. Toutes les fourmis ne participent pas à la recherche de la nourriture, celles restant au nid s'occupent principalement du couvain. Le comportement de recherche de nourriture est individuel. Par exemple, les ouvrières ne déposent pas de message chimique (traces de phéromone) sur le sol pour indiquer à d'autres fourrageuses le chemin menant à une source de nourriture. Elles ne mettent donc pas en œuvre des comportements de recrutement de masse.

Nous pouvons supposer que la nature des proies recherchées n'encourage pas spécialement ce genre de communication inter individus : la présence des proies étant aléatoire. Sa capture d'une proie ne donne que peu d'informations sur la stabilité spatiale de la source de nourriture. Autrement dit, la probabilité de retrouver une proie dans la même zone qu'une précédente capture n'est pas suffisante pour y canaliser les forces de fourragement de la colonie. Cependant, individuellement, les ouvrières mémorisent leur site de capture et lors de leur prochaine sortie du nid, elles retournent systématiquement sur le dernier site de chasse fructueux. Cette spécialisation sectorielle est une réponse pour l'adaptation nécessaire à la découverte et l'exploitation de sources de nourriture. Ce type de fourragement solitaire se retrouve particulièrement chez les espèces peu populeuses. Les espèces à population importante utilisent des mécanismes de recrutement massif beaucoup plus couramment [31]. Les fourrageuses solitaires développent en conséquence des mécanismes d'apprentissage plus évolués.

De sortie en sortie, les ouvrières s'éloignent du nid car la probabilité de trouver une proie est inversement proportionnelle à la densité de fourrageuses qui décroit évidemment quand la distance du nid augmente. Ainsi, les fourrageuses ont un comportement collectif indirect puisque de manière statistique elles coopèrent

pour couvrir au mieux leur espace de recherche que constitue le voisinage du nid. Elles construisent de cette façon une mosaïque de zones de chasse qui couvre la périphérie du nid. La figure 2-1 présente les aires de fourragement d'une colonie *Pachycondyla Apicalis.*

Le comportement de fourragement de *Pachycondyla Apicalis* peut être résumé en trois règles essentielles :

1. La découverte d'une proie entraîne toujours le retour sur le site lors de la sortie suivante. C'est là que la fourrageuse reprend ses nouvelles prospections.
2. La découverte d'une proie pèse sur la décision de sortie des fourrageuses en réduisant l'intervalle de temps qu'elles passent au nid.
3. Les fourrageuses semblent progressivement apprendre une association entre une direction de recherche opposée au nid et l'augmentation de la probabilité de succès.

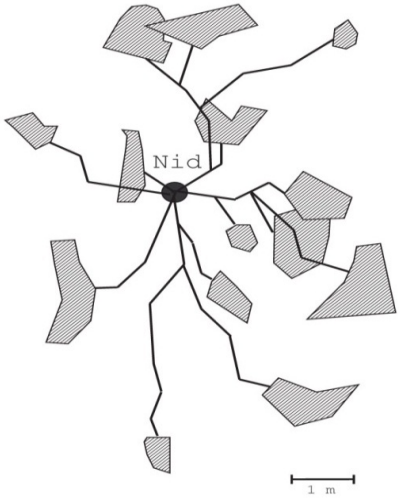


Figure 2.1 : Exemple (fictif) de carte des trajets et aires de récolte des fourrageuses.

Ces trois règles ont l'avantage d'être simples, on peut cependant les renforcer par quelques points importants pour la mise en œuvre de l'algorithme :

- Lors de ses premières sorties, la fourmi choisit une direction aléatoire, s'éloigne peu du nid et retourne à l'abri de celui-ci à la moindre alerte.

- Si la fourmi capture une proie, elle retourne directement au nid et mémorise visuellement le chemin.
- Le retour sur un site de chasse se soldant par un échec peut se produire plusieurs fois de suite.
- Un site de chasse ne produisant plus le renforcement que constitue la capture de proie, est abandonné mais pas obligatoirement oublié par la fourrageuse.
- L'exploration d'un site de capture privilégie les directions qui éloignent la fourmi du nid dans une limite de périmètre imposé par le cout énergétique prohibitif que représente un échec à une grande distance du nid.
- Le déménagement du nid sur la position du meilleur site de chasse a pour effet de réinitialiser la mémoire des fourrageuses.

Comme nous l'avons déjà mentionné, la fragilité du nid impose des déménagements réguliers. Des fourmis éclaireuses partent alors à la recherche d'un nouvel abri. Le déménagement s'effectue grâce à des mécanismes de *recrutement en tandem* où une fourmi se fait guider par une de ses congénères jusqu'au nouvel emplacement. Ce changement de nid, a pour effet de réinitialiser la mémoire des fourrageuses. C'est à cette occasion que l'utilisation de repères visuels prend toute son importance : le marquage de chemins avec phéromones perturberait l'adaptation des fourmis à la situation de leur nouveau nid, car les anciens chemins interféraient avec les nouveaux. Avec une mémoire visuelle, les fourrageuses reconstruisent un réseau de sites de chasse autour du nouveau nid plus aisément, plus rapidement. On peut donc parler d'apprentissage en ce qui concerne la recherche de proies : la fourmi apprend les sites qui lui rapportent de la nourriture et est capable de les oublier quand elle n'y trouve plus de proie ou que le nid a changé de localisation.

L'aspect individuel de la recherche est particulièrement adapté à la fréquence d'apparition des proies : si une proie tombe sur le sol et qu'elle est capturée, la fourrageuse reviendra explorer le site toute seule. Si la proie est trop lourde, elle sera découpée puis transportée en plusieurs voyages par la même fourmi qui utilise de cette façon sa mémoire. Si le site de chasse se trouve être un gisement (par exemple de termites), la fourrageuse reviendra systématiquement jusqu'à épuisement du site.

L'intérêt de la stratégie de fourragement des fourmis *Pachycondyla Apicalis* réside dans sa simplicité et la bonne couverture de l'espace de recherche qui en résulte. Nous avons ici un phénomène d'émergence de règles de recherche simples et individuelles, qui ne tiennent pas compte du travail des autres fourmis. Nous obtenons une exploration radiale de l'espace centrée sur le nid.

Dans la réalité, l'efficacité de la stratégie n'est pas parfaite si on considère le nombre de proies trouvées relativement au nombre de proies présentes [32]. Cependant, la taille d'une colonie de *Pachycondyla Apicalis* étant relativement réduite, les besoins en nourriture, sont modestes. Il semble que l'adaptation de ces fourmis ne réside pas seulement dans leur comportement de fourragement mais aussi dans le maintien de colonies peu populeuses. Ceci permet aux *Pachycondyla Apicalis* de survivre dans des secteurs comportant de nombreux prédateurs concurrents et de ne pas nécessiter une grande quantité d'insectes pour se nourrir.

2-3. Modélisation algorithmique

A partir de la biologie des *Pachycondyla Apicalis* qui a été présentée en détail dans le paragraphe précédent, nous présenterons dans ce qui suit l'adaptation de la stratégie de fourragement des *Pachycondyla Apicalis* à la résolution de problèmes d'optimisation [3,18]. Nous confirmerons la généricité de la méthode relativement à l'espace de recherche avant d'étudier l'influence des différents paramètres sur le rendement de l'algorithme déduit.

2-3-1. Espace de recherche et fonction d'évaluation :

On considère une population de n fourmis fourrageuses $a_1,\ldots\ldots,a_n$ de l'espèce *Pachycondyla Apicalis.* Ces agents sont positionnés dans l'espace de recherche noté S, et vont tenter de minimiser une fonction d'évaluation f définie de S dans R.

Chaque point $s \in S$ est une solution valide du problème, ce qui signifie que f est définie en tout point de S. Cet espace de recherche peut être un espace continu,

binaire ou un espace de permutations. L'algorithme **API** est générique en ce qui concerne l'espace de recherche S, c'est là l'un de ses principaux atouts. La définition des deux opérateurs suivants est suffisante pour déterminer le déplacement des fourmis :

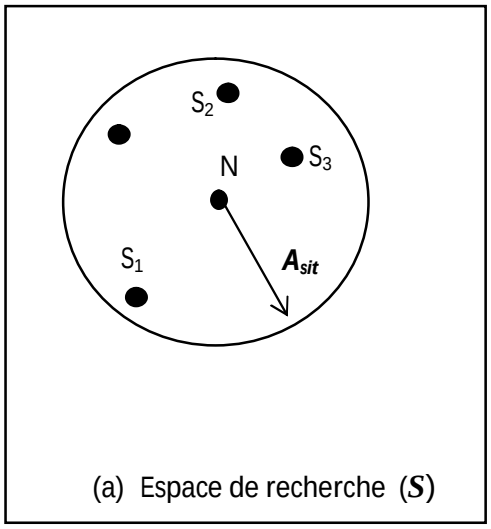
- L'opérateur O_{rand} qui génère un point s de S de manière uniformément aléatoire.

- L'opérateur O_{explo} qui génère un point s' dans le voisinage du point s.

Concernant l'opérateur O_{explo} , la taille du voisinage de s est paramétrée par une amplitude, notée A telle que $A \in [0,1]$. Cette amplitude fixe la portée de l'exploration autour de s relativement à la taille de l'espace. O_{explo} peut être une exploration aléatoire tout comme une heuristique inspirée par le domaine de recherche.

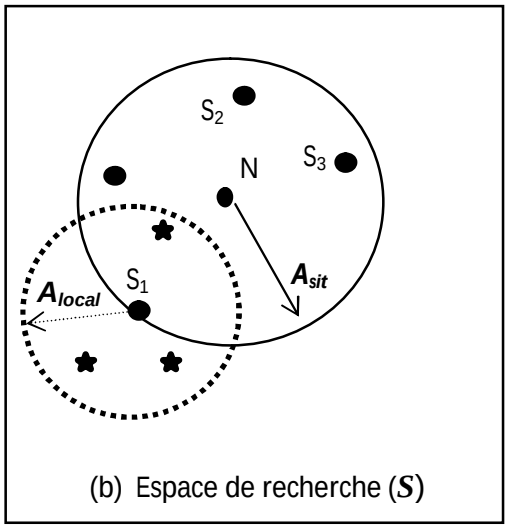
2-3-2. Comportement local des fourmis

Après chaque déplacement du nid **N,** la fourrageuse formule une nouvelle liste de p sites de chasse qu'elle mémorise (figure 2-2), un site de chasse est un point s de l'espace de recherche S construit par un opérateur O_{explo} utilisant un rayon de recherche A_{sit} autour du nid **N**. Puis la fourmi va commencer une exploration locale au voisinage des p sites de chasse mémorisés (figure 2-3).



(a) Espace de recherche (S)

Figure 2.2 : Recherche de sites de chasse. La fourmi se constitue une liste de p=4 sites de chasse au voisinage du nid N à une distance maximale A_{sit}



(b) Espace de recherche (S)

Figure 2.3 : Exploration locale de la fourmi autour du site S_1. Les étoiles représentent les explorations menées dans la zone de rayon A_{local}.

Initialement, quand l'intérêt des sites est inconnu, la fourmi choisit un site ***s*** au hasard parmi les ***p*** dont elle dispose. L'exploration locale consiste à construire un point ***s'*** de l'espace de recherche grâce à l'opérateur O_{explo} avec un rayon de recherche A_{local}. La fourrageuse capture une proie si cette exploration locale conduit à une meilleure valeur de ***f***, soit : ***f(s)*** < ***f (s')***. A chaque fois qu'une fourmi parvient à améliorer ***f(s)***, elle mémorise le site ***s'*** à la place de ***s*** et sa prochaine exploration locale se fera dans le voisinage de ***s'***.

Si l'exploration locale est infructueuse, la fourmi choisira pour une exploration future, un site au hasard parmi les ***p*** sites qu'elle a en mémoire.

Quand un site a été exploré plus de P_{locale} fois sans succès, il est définitivement oublié et sera remplacé par un nouveau site à la prochaine sortie (prochaine itération). Le paramètre P_{locale} représente une patience locale.

L'organigramme de la figure 2-4 résume le comportement individuel d'une fourrageuse.

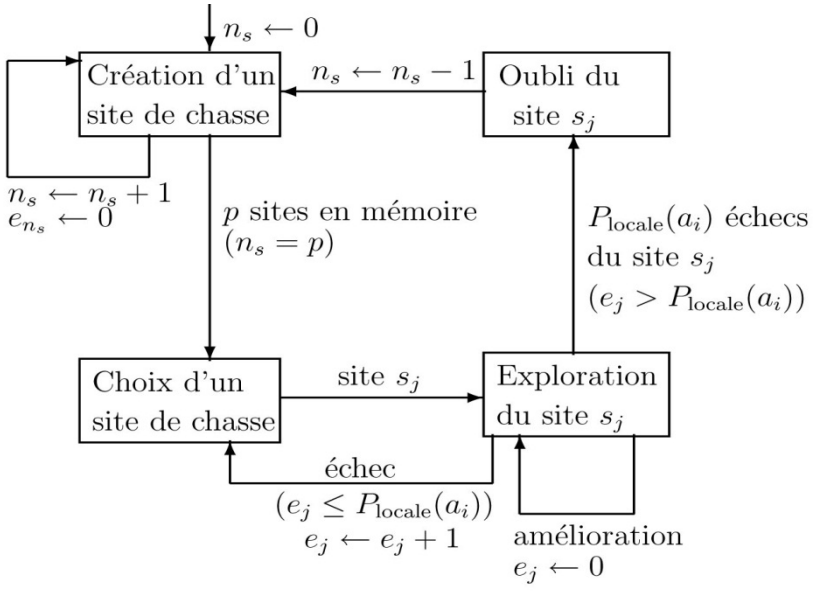


Figure 2.4 Organigramme résumant le comportement individuel d'une fourrageuse.

2-3-3. Exploration globale de l'espace de recherche

D'un point de vue global, **API** place aléatoirement le nid à une position **N** de l'espace S, et commence l'exploration de l'espace au voisinage de **N**. L'exploration est déterminée par le comportement local des fourmis. A chaque pas de l'algorithme, les ***n*** fourmis sont simulées en parallèle. A l'initialisation, le nid est placé dans l'espace de recherche S d'une manière uniformément aléatoire par l'opérateur O_{rand}. Puis le nid est déplacé tous les P_N déplacements des ***n*** fourmis. Il est alors placé sur le meilleur point s^+ trouvé depuis son dernier emplacement. A chaque déplacement du nid, les fourmis reprennent leur exploration à partir de la nouvelle position du nid. La figure 2-5 montre la stratégie globale de l'exploration. P_N représente un paramètre de patience pour le nid. Ce paramètre est fixé suivant la patience locale d'une fourmi P_{local} et la taille de sa mémoire ***p***.

$$p_N = 2 \times (p_{local} + 1) \times p \qquad (2.1)$$

Ce calcul a pour but de laisser suffisamment d'itérations entre chaque déplacement de nid pour que les fourmis puissent créer et explorer leurs ***p*** sites de chasse. Une autre solution serait de ne déplacer le nid que si l'optimum n'a pas été amélioré depuis un certain nombre d'itérations P_N.

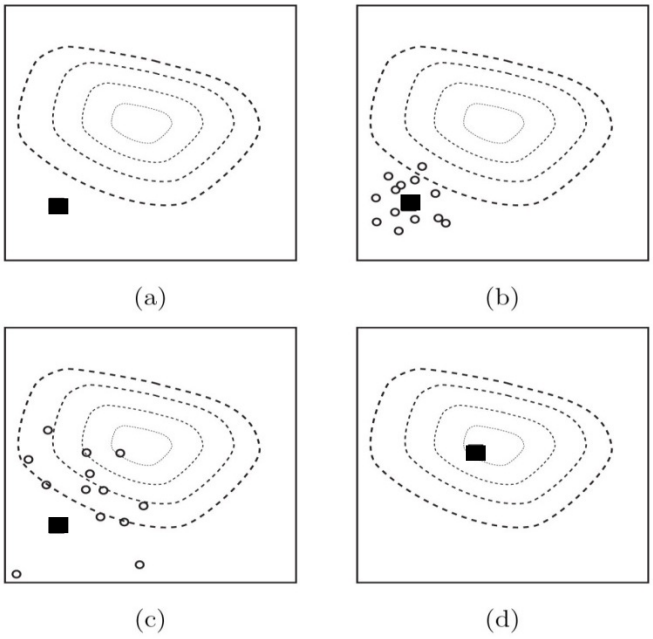


Figure 2.5 : ***Exploration globale et déplacement du nid.*** En (a), le nid (carré) est placé aléatoirement dans l'espace de recherche. En (b), sont représentés les sites de chasse (cercles) crées autour du nid. En (c), à cause de l'exploration locale, les sites de chasse se déplacent vers des zones plus intéressantes de l'espace de recherche (les pointillés les plus au centre). En (d), le nid est déplacé sur la position du meilleur site de chasse, les sites seront ensuite générés à partir de cette nouvelle position, comme en (b), et ainsi de suite.

Enfin, à chaque déplacement du nid, la mémoire des fourmis est vidée et les fourmis doivent reconstruire leurs ***p*** sites de chasse. Du point de vue de l'optimisation, cela permet d'éviter des minima locaux dans lesquels les fourmis resteraient enfermées. Cela permet aussi de rassembler les fourmis autour du meilleur point trouvé et ainsi de concentrer les recherches. On pourrait cependant procéder d'une manière plus douce : il suffirait de placer le nid à la position du minimum global trouvé par la colonie à chaque fois que celui-ci est amélioré sans réinitialiser toutes les fourmis. Ainsi, quand une fourmi crée un nouveau site de chasse, elle le ferait dans le voisinage de l'optimum global. On se débarrasse alors du choix de la patience du nid.

2-3-4. Algorithmes

Les principales étapes de la simulation de la colonie de fourmis *Pachycondyla Apicalis* sont données par l'algorithme suivant [3]:

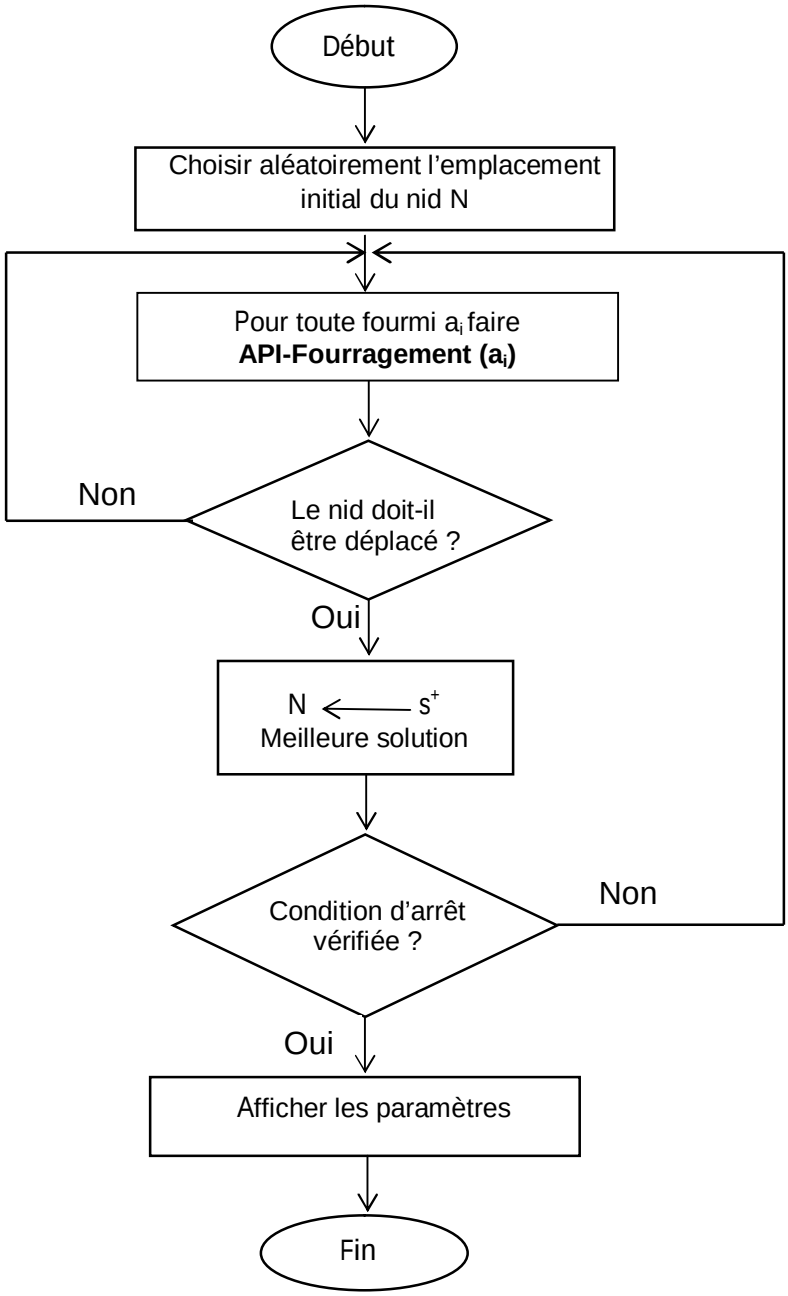


Figure 2.6 : « Algorithme **API**»,Simulation d'une colonie de fourmis *Pachycondyla apicalis.*

Le comportement local de fourragement d'une fourmi $\boldsymbol{a_i}$ est donné par l'algorithme suivant [3]:

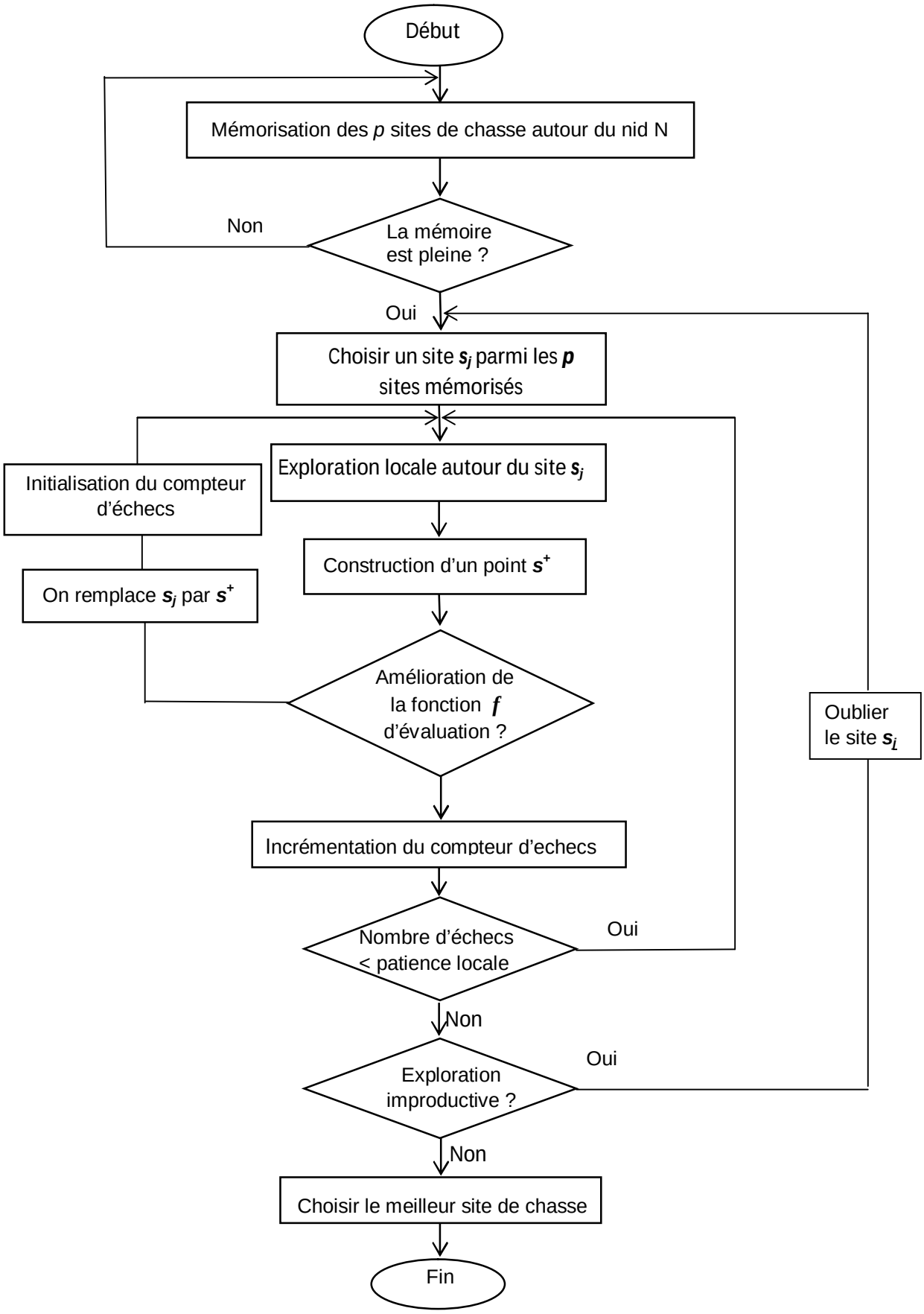


Figure 2.7 :« Algorithme **API- Fourragement** (a_i) » ; comportement local d'une fourmi a_i.

2-4. Extensions de l'algorithme API

Afin d'enrichir l'algorithme API, quelques inspirations biologiques supplémentaires ont été exploitées, elles seront présentées dans ce qui suit :

2-4-1. Recrutement

Comme vu précédemment, *Pachycondyla Apicalis* ne faisait pas de recrutement pendant le fourragement, cependant, il est possible d'introduire le recrutement en *tendem running* (marche en arrière) dans API [34]. Ainsi, une fourmi qui parvient à améliorer un site, tente de recruter une autre fourmi en lui transmettant la position de son site de chasse. Si elle y parvient, le nombre de recherches locales pour ce site va augmenter. Un paramètre $\boldsymbol{P_{recrut}}$ représentant la probabilité pour qu'une fourmi tentant un recrutement y parvienne, sera introduit, et nous noterons, par la suite, l'algorithme API avec recrutement, l'algorithme $\mathbf{API_r}$.

2-4-2. Population hétérogène

Afin de choisir les paramètres de l'algorithme API ainsi défini, deux propositions ont été envisagées :

- Variation des paramètres au cours du temps. Nous avons pu constater dans la nature, que les fourmis n'ont pas la même stratégie de chasse en fonction de leurs âges, plus elles sont âgées, plus elles sortent fréquemment du nid et plus elles vont chasser loin de celui-ci [30]. Une autre façon de voir les choses, serait d'étudier l'environnement, par exemple à travers les variations de température. En effet, l'activité des ouvrières est liée à la température extérieure. Par exemple, les *Pachycondyla Apicalis* chassent plus aux heures chaudes de la journée [8]. Du point de vue de la modélisation, on pourrait traduire ces deux variations : d'âge et de température par une variation des paramètres de chaque fourmi au cours du temps.

- Hétérogénéité des paramètres. Dans la nature, les fourmis ont des caractéristiques différentes les unes des autres, elles sont parfois même regroupées en castes du fait de leurs différences morphologiques. Des modèles de division du travail ont été élaborés pour rendre compte de la capacité dynamique des fourmis à effectuer certaines tâches [33]. Cette plasticité nous suggère de constituer une population de fourmis ayant des caractéristiques variées.

Nous utiliserons cependant, un paramétrage statique car la variation d'un paramètre au cours du temps est délicate surtout si elle est adaptative. La population de fourmis sera donc constituée d'individus ayant des paramètres différents. Par la suite, les populations hétérogènes utilisées dans API que nous noterons $API_{h,}$ se composeront de fourmis dont seules les amplitudes $\boldsymbol{A_{sit}}$ et $\boldsymbol{A_{local}}$ varieront d'une fourmi à une autre et seront fixées dans le temps. On notera, cependant, qu'une supervision par un algorithme évolutionnaire est très envisageable et peut être utilisée pour fixer les amplitudes $\boldsymbol{A_{sit}}$ et $\boldsymbol{A_{local}}$ de chaque fourmi. Nous fixerons les valeurs des deux amplitudes de façon automatique afin de couvrir le plus de combinaisons possibles :

$$A_{sit}(a_1)=A.\varepsilon^{0},\ldots..,\ A_{sit}(a_i)=A.\varepsilon^{i-1},\ldots\ldots,\ A_{sit}(a_n)=A.\varepsilon^{n-1}. \qquad (2\text{-}2)$$

Où $\varepsilon = \frac{1}{0.01}^{\frac{1}{n-1}}$ et A est l'amplitude de recherche relative à la première fourmi.

Le paramètre $\boldsymbol{A_{local}(a_i)}$ est fixée pour toutes les fourmis à $0.1.A_{sit}(a_i)$.

La figure ci-dessous donne en exemple les valeurs des paramètres $\boldsymbol{A_{sit}}$ et $\boldsymbol{A_{local}}$ pour 10 fourmis :

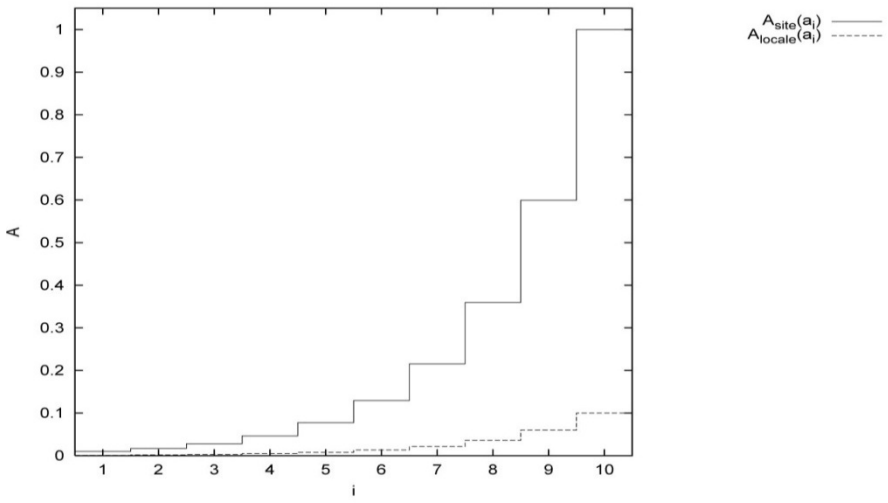


Figure 2-6 : Valeurs des paramètres $\boldsymbol{A_{sit}}$ et $\boldsymbol{A_{local}}$ pour 10 fourmis.

2-4-3. Prise en compte de la décision de sortir du nid

Un certain nombre de choix ont été arbitrairement faits pour la modélisation des *Pachycondyla Apicalis.* Nous n'avons en effet pas pris en compte certaines

caractéristiques de cette espèce qui peuvent cependant se révéler intéressantes du pour notre problème d'optimisation. Le comportement prédateur d'une fourmi évolue par exemple en fonction de son âge [30]. Les jeunes fourrageuses (naïves) ont tendance à rester près du nid sans présenter de spécialisation spatiale alors que les fourrageuses expérimentées s'éloignent beaucoup plus du nid tout en montrant une fidélité à une zone de chasse importante. En ce qui concerne la recherche d'un optimum, cela consiste à augmenter l'effort de recherche dans le voisinage du point central. Bien que nous n'ayons pas utilisé explicitement de fourmis naïves, la diversité statique introduite dans les paramètres dans le cas d'une population hétérogène, reproduit en quelque sorte cette diversification basée sur l'âge.

Un deuxième aspect n'a pas été pris en compte : l'intervalle de temps qui sépare deux sorties est plus court si la fourmi a rencontré un succès à sa première sortie. S'il est probable que ce comportement permet une certaine économie d'énergie au niveau de la colonie en n'allouant des essais qu'aux fourmis les plus chanceuses/efficaces, il n'est pas évident que dans la transposition au problème de l'optimisation cela améliore les performances de l'ensemble. Comme nous considérons que l'évaluation de la fonction objectif représente une contrainte de ressource, nous pouvons proposer une amélioration qui va dans le sens de relier la décision de sortir du nid d'une fourmi $\boldsymbol{a_i}$ à une probabilité $\boldsymbol{P_s(a_i)}$ qui sera modifiée, à chaque itération, de la manière suivante :

$$P_s(a_i) \longleftarrow (1-\alpha).\ P_s(a_i) + \alpha.\ \delta \qquad (2\text{-}3)$$

Où $\boldsymbol{\alpha}$ est un paramètre réel compris dans l'intervalle [0,1] et commun à toutes les fourmis. $\boldsymbol{\delta}$ est une valeur binaire qui vaut 1 si la dernière sortie a été fructueuse et 0 sinon. Au départ, et à chaque déplacement du nid, on fixera $\boldsymbol{P_s}$ à 1 et $\boldsymbol{\delta}$ à 0.

2-5. Les paramètres d'API

Avant toute application de l'algorithme API, on se doit de fixer les valeurs des différents paramètres qui entrent dans la composition de l'algorithme. Le tableau 2.1 nous donne une liste récapitulative de ces paramètres.

Paramètres	***Descriptions***
n	nombre de fourmis
A_{sit}	amplitude maximale de création d'un site
A_{local}	amplitude d'exploration locale
P_{local}	patience locale d'une fourmi
P_N	patience du nid
p	nombre de sites mémorisés par une fourmi
P_{recrut}	probabilité de recrutement
O_{rand}	opérateur générant aléatoirement un point s de l'espace
O_{explo}	opérateur générant un point s' au voisinage de s

Tableau 2.1. Tableau récapitulatif des paramètres d'API.

2.6. Conclusion

Dans ce chapitre, nous avons exposé l'algorithme de colonies de fourmis **API** déduit de la modélisation du comportement de fourragement de l'espèce *Pachycondyla apicalis* et son application au problème général de l'optimisation.

Cet algorithme présente l'avantage d'être relativement simple à décrire. Sa description est indépendante de l'espace de recherche considéré dans l'application, contrairement aux autres algorithmes de colonies de fourmis que nous avons brièvement décrits précédemment.

Ceci encourage l'utilisation de l'algorithme **API** pour la résolution d'une multitude de problèmes et ceci dans divers domaines. Nous pensons principalement aux problèmes d'optimisation continue difficile. Malgré, malheureusement dans la

littérature, très peu de travaux ont été faits dans ce sens, et les quelques applications qu'on rencontre se résument à de simples tests élaborés pour l'estimation des performances de l'**API** ou de l'une de ses variantes.

CHAPITRE 3

ETUDE DES FILTRES NUMERIQUES

3.1. Introduction

On appelle « **filtre numérique** » un système utilisé pour modifier la distribution fréquentielle d'un signal numérique selon des spécifications données. Un filtre numérique peut être vu comme un procédé de calcul permettant de transformer un signal numérique d'entrée (séquence de nombres) en un signal numérique de sortie (seconde séquence de nombres) pour obtenir la modification voulue du signal. Le problème du filtrage numérique consiste donc à déterminer l'équation régissant cette transformation des signaux numériques qui d'une part doit représenter la réponse fréquentielle spécifiée et d'autre part peut être effectivement réalisée. La transformation peut être implantée sous forme de logiciel (algorithme) ou matériel (circuits électroniques).

Les filtres numériques sont, pour les signaux échantillonnés, les équivalents des filtres analogiques pour les signaux continus. En raison du développement des circuits intégrés rapides, les filtres numériques deviennent plus intéressants que les filtres analogiques en apportant de nombreux avantages : précision, fiabilité, stabilité, adaptabilité et facilité de commande.

De la même manière, le problème consiste à réaliser un filtre donnant une réponse fréquentielle $H(f)$ donnée (prédéfinie à l'aide d'un gabarit), une réponse impulsionnelle $h(t)$ fixée ou éventuellement une réponse indicielle voulue.

Dans le cas général de ces filtres, la valeur de la sortie numérique $y(kTe) = y_k$ à l'instant kTe est fonction de l'entrée $x(kTe) = x_k$ au même instant kTe et des N entrées numériques précédentes $x(iTe) = x_i$ pour tout $i \in \{k - 1, \ldots, k - N\}$ et de plus des sorties numériques précédentes $y(jTe) = y_j$ pour tout $j \in \{k - 1, \ldots, k - N\}$.

Cela impose que la fonction générale donnant les échantillons de sortie y_k n'est autre qu'une combinaison linéaire des échantillons x_i et y_i [35].

D'où :

$$y_k = \sum_{q=0}^{Q} a_q \; x_{k-q} - \sum_{m=1}^{M} b_m \; y_{k-m} \qquad (3.1)$$

En passant au domaine fréquentiel, moyennant la transformée en z, on déduit de l'équation (3.1), l'expression de la fonction de transfert, qui elle aussi permet de définir un filtre numérique.

Ainsi, la fonction de transfert générale d'ordre Q d'un filtre numérique est la suivante [36].

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{m=0}^{M-1} b_m \; z^{-m}}{1+\sum_{q=1}^{Q} a_q \; z^{-q}} \qquad (3.2)$$

Rappelons que la réponse en fréquence s'obtient en remplaçant **z** par

$$\exp(\pi j f_n) = \cos \pi f_n + j \sin \pi f_n \qquad (3.3)$$

Avec $f_n = 2\,(f/f_e)$, f_e étant la fréquence d'échantillonnage.

3.2. Stabilité des filtres numériques

La stabilité des filtres numériques est un critère de grande importance. On dira qu'un filtre numérique set stable, si à toute entrée x_n bornée par la valeur M correspond une sortie y_n bornée par une valeur M' [37].

$$\exists\, M \quad \forall |x_n| < M \Longrightarrow \qquad \exists\, M' \quad \forall |y_n| < M'$$

Il existe par conséquent, deux conditions nécessaires et suffisantes, pour qu'un filtre numérique soit stable.

- **1ère condition nécessaire et suffisante de stabilité**

  Il faut que les échantillons de la réponse impulsionnelle soient bornés par une valeur A. Ceci est traduit par l'inégalité suivante

$$\sum_{n=-\infty}^{+\infty} |h(n)| < A$$

- **2ème condition nécessaire et suffisante de stabilité**

  Lorsque le filtre est causal et que sa fonction de transfert est rationnelle, la condition est qu'il faut et il suffit que tous les pôles de H(z) soient à l'intérieur du cercle unité du plan z.

3.3. Filtres à réponse impulsionnelle finie RIF

Les filtres numériques à réponse impulsionnelle finie (RIF) sont des systèmes linéaires et invariants dans le temps définis par une équation selon laquelle un nombre de sortie, représentant un échantillon du signal sortant filtré, est obtenu par sommation pondérée d'un ensemble fini de nombres d'entrée représentant les échantillons du signal d'entrée [36]. La relation reliant les échantillons de sortie aux échantillons d'entrée est donnée par :

$$y_k = \sum_{m=0}^{M-1} b_m \,.\, x(k-m) \qquad (3.4)$$

Du fait qu'un échantillon de sortie ne dépend que des échantillons des entrées précédentes, ce filtre est appelé aussi **filtre non récursif**.

Un filtre ainsi défini comporte un nombre *N* fini de coefficients b_i. Sa réponse impulsionnelle est la suite *h(i)* définie comme suit :

$$h(i) = \begin{cases} b_i & 0 \le i \le N-1 \\ 0 & ailleurs \end{cases} \qquad (3.5)$$

La fonction de transfert en z d'un filtre numérique RIF est donnée par l'équation suivante :

$$H(z) = \sum_{m=0}^{M-1} b_m \, z^{-m} \qquad (3.6)$$

3.3.1. Structures de réalisation d'un filtre RIF

A partir de la fonction de transfert *H*(*z*), obtenue selon les différentes transformations possibles, diverses structures peuvent être utilisées : structure **directe** (implémentation de l'équation aux différences), structure **canonique** (structure directe avec minimisation des composants) et structure **en éléments simples**.

La figure ci-dessous schématise sa structure directe [38] :

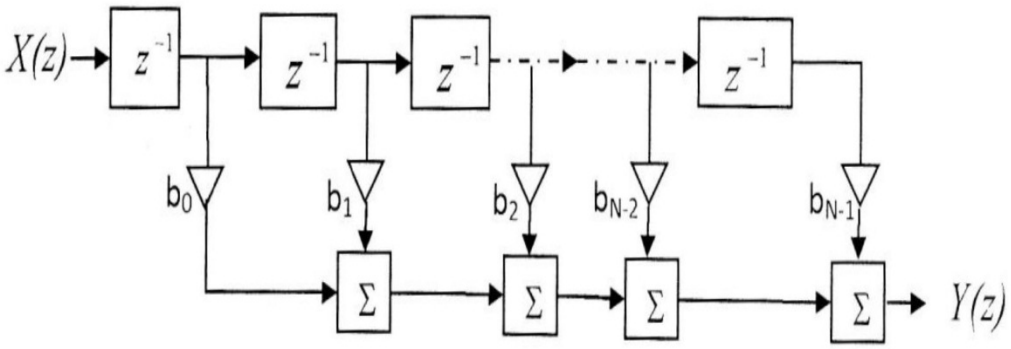


Figure 3.1 : Structure directe d'un filtre numérique RIF

3.3.2. Stabilité des filtres RIF

La fonction de transfert d'un filtre numérique RIF ne possède pas de pôles (uniquement des zéros). Par conséquent et d'après le paragraphe 3.2, le filtre RIF est inconditionnellement stable.

3.3.3. Synthèse des filtres RIF

Les filtres RIF permettent de réaliser le filtrage numérique au moyen du produit de convolution défini par la relation 3.4. Les valeurs dd sortie *y(i)* sont obtenues par une somme pondérée d'un ensemble fini de valeurs d'entrée *x(i).*

Quant au calcul coefficients de pondération *h(i),* plusieurs méthodes ont été proposées dans la littérature [39,40].

Ces méthodes développées sous forme d'algorithmes de synthèse, comportent en général quatre étapes :

- Choix du filtre désiré, souvent donné par son gabarit comportant ses caractéristiques fréquentielles.

- Choix du type de filtre à concevoir, RIF ou RII.

Choix d'un critère d'évaluation du filtre à obtenir par rapport au filtre désiré.

- Développement d'une méthode pour obtenir le filtre optimal répondant aux critères désirés.

D'un point de vue général, ces méthodes reposent sur des techniques pouvant être regroupées en trois classes :

- Technique de fenêtrage [49].
- Technique de l'échantillonnage en fréquence [49].
- Techniques d'optimisation. Parmi lesquelles nous pouvons citer la méthode des moindres carrés [41,42,43], et la méthode de Parks-Maclellan, qui n'est autre qu'une variante de l'algorithme de Remez appliqué à la synthèse des filtres RIF [44,45].

3.4. Filtres à réponse impulsionnelle infinie RII

Les filtres numériques à réponse impulsionnelle infinie sont des systèmes linaires invariants dans le temps, décrits par la relation entrée-sortie [37], donnée par:

$$y_k = \sum_{m=0}^{M-1} b_m \cdot x(k-m) - \sum_{q=0}^{Q-1} a_q \; y(k-q) \qquad (3.7)$$

Ces filtres sont appelés aussi filtres ***récursifs***, ont une fonction de transfert rationnelle, dont la transformée en z est donnée par :

$$H(z) = \frac{\sum_{m=0}^{M-1} b_m .z^{-m}}{\sum_{q=0}^{Q-1} a_q \ z^{-q}} \qquad (3.8)$$

Avec $Q \geq M$ (souvent, en prend $Q = M = N$, N étant l'ordre du filtre).

3.4.1 Structures de réalisation d'un filtre RII

3.4.1.1 Structure directe

La structure de réalisation directe d'un filtre RIF est obtenue par La traduction directe de la transformée en z de sa fonction de transfert [36].

Cependant, on mettant

$$H(z) = \frac{Y(z)}{R(z)} \cdot \frac{R(z)}{X(z)} = H_1(z) \cdot H_2(z) \qquad (3.9)$$

On peut obtenir deux types de structures :

Structure directe type I, en mettant

$$H_1(z) = \sum_{i=0}^{N} b_i \ z^{-i} \qquad et \qquad H_2(z) = \frac{1}{1+\sum_{j=1}^{N} a_j \cdot z^{-j}} \qquad (3.10)$$

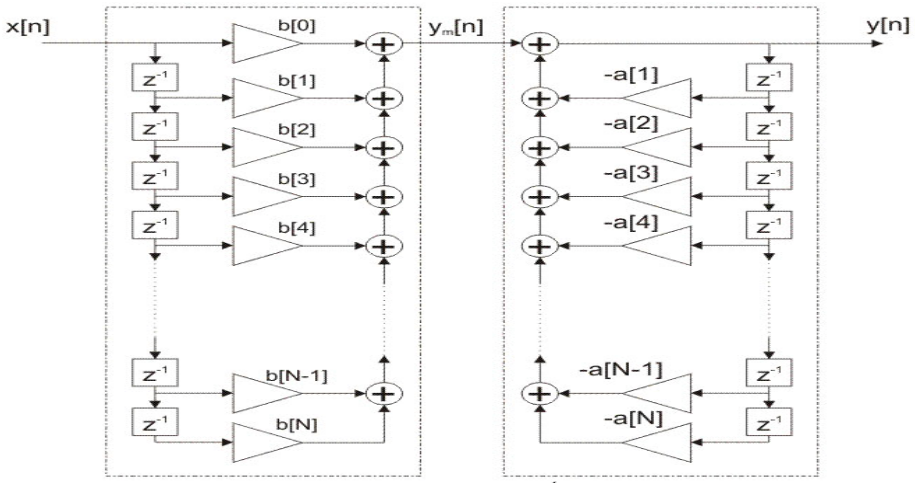


Figure 3.2 : Structure directe type I d'un filtre numérique RII

Structure directe type II, en mettant

$$H_2(z) = \sum_{i=0}^{N} b_i\, z^{-i} \qquad et \qquad H_1(z) = \frac{1}{1+\sum_{j=1}^{N} a_j \cdot z^{-j}} \tag{3.11}$$

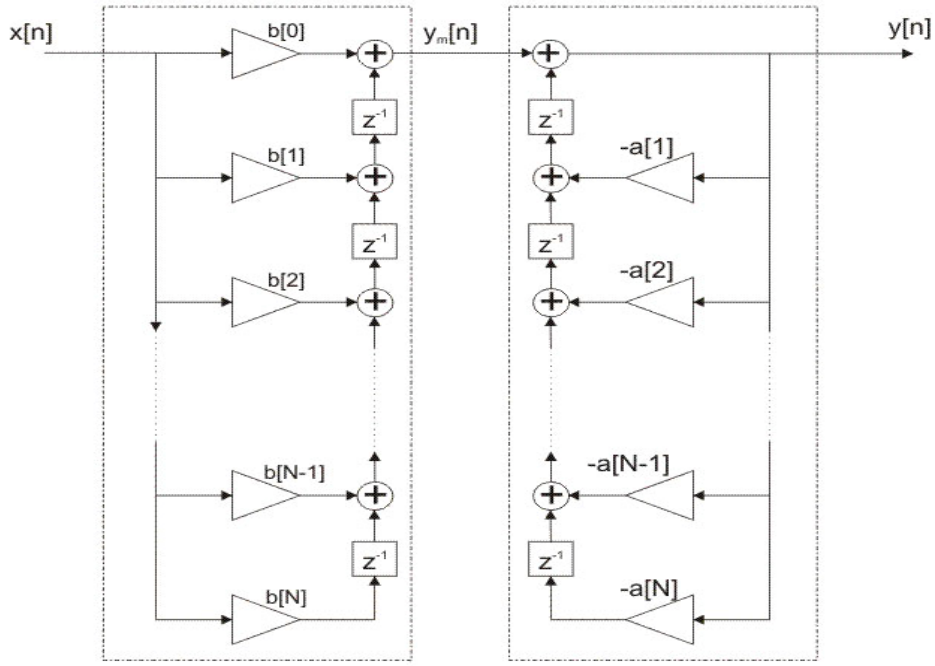


Figure 3.3 : Structure directe type II d'un filtre numérique RII

3.4.1.2. Structure décomposée

La structure directe d'un filtre numérique RII est obtenue en effectuant une décomposition en somme (structure en parallèle) ou en produit (structure en cascade) des fonctions élémentaires du premier ou du second ordre réalisés séparément [36]. On obtient ainsi :

Structure en parallèle

$$H(z) = K_1 + \sum_{i=1}^{r} H_i(z) \tag{3.12}$$

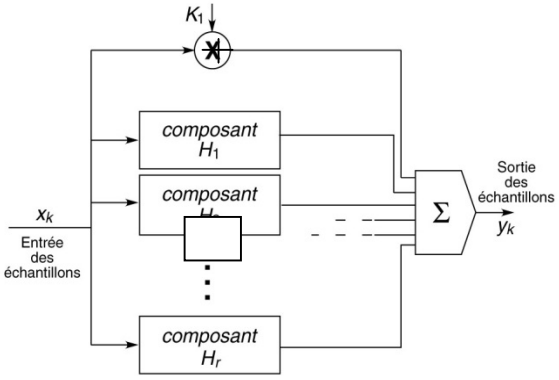


Figure 3.4 : Structure en parallèle d'un filtre numérique RII

Structure en Cascade

$$H(z) = K_2 \cdot \prod_{j=1}^{r} H_i(z) \qquad (3.13)$$

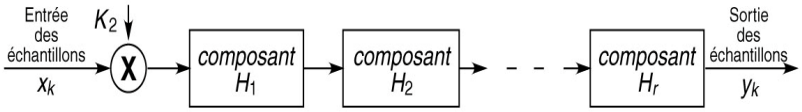


Figure 3.5 : Structure en cascade d'un filtre numérique RII

3.4.1.3. Cellules du premier ordre

La réponse impulsionnelle d'une cellule du 1er ordre est donnée par relation de récurrence suivante :

$$y(n) = x(n) - a \,.\, y(n) \quad , a \epsilon R \qquad (3.14)$$

D'où la structure suivante :

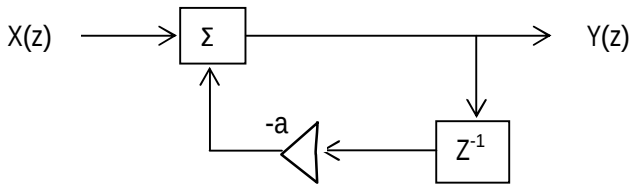


Figure 3.6 : Structure d'une cellule du 1er ordre

A partir de la relation (3.14) nous obtenons la fonction de transfert suivante :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1+a\,.\,z^{-1}} = \frac{z}{z+a} \qquad (3.15)$$

La condition de stabilité est vérifiée si tous les pôles se trouvent à l'intérieur du cercle unité, c'est-à-dire |a|<1.

3.4.1.4. Cellules du second ordre

La réponse impulsionnelle d'une cellule du 2ème ordre est donnée par relation de récurrence suivante :

$$y(n) = x(n) - a_1\,.y(n-1) - a_2.y(n-2) \quad ,a\epsilon\, R \qquad (3.16)$$

D'où la structure suivante :

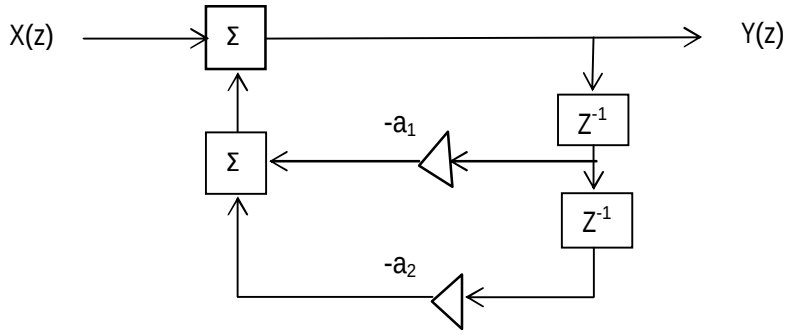


Figure 3.7: Structure d'une cellule du second ordre

La fonction de transfert en z d'une telle cellule est donnée par :

$$H(z) = \frac{1}{1+a_1.\,z^{-1}+a_2\,z^{-2}} = \frac{z^2}{z^2+a_1.\,z^1+a_2} \qquad (3.17)$$

Cette fonction possède, suivant le signe de $a_1{}^2 - 4a_2$ des pôles réels ou complexes conjugués :

$a_1^2 - 4\,a_2 \geq 0$: La fonction possède deux pôles réels, leurs images, sont situés sur l'axe des réels. La fonction de transfert est alors le produit de deux fonctions du premier ordre, ces fonctions sont identiques si $a_1{}^2 = 4.a_2$.

$a_1^2 - 4\,a_2 < 0$: Les deux pôles sont complexes conjugués. Leurs images sont *p* et *p** telle que :

$$z_p = \frac{1}{2}\,(-a_1 \mp j\sqrt{4 \cdot a_2 - a_1{}^2} \qquad (3.18)$$

3.4.2. Stabilité des filtres numériques RII

Pour que le filtre soit stable, il faut que le module de ses pôles soit inférieur à 1. Autrement dit, à l'intérieur du cercle unité du plan z [38].

Cas des pôles réels

La condition de stabilité est que

$$\frac{1}{2}\left(-a_1 + \sqrt{a_1{}^2 - 4.a_2}\right) < 1 \quad et \quad \frac{1}{2}\left(-a_1 - \sqrt{a_1{}^2 - 4.a_2}\right) > -1 \qquad (3.19)$$

D'où : $\left|a_1\right| < 1 + a_2$ (3.20)

Cas des pôles complexes

Dans ce cas, la condition de stabilité est que $a_2 < 1$.

Le domaine de stabilité est donc un triangle délimité par trois droites d'équations :

$$a_2 = 1, \quad a_2 = -1 - a_1, \quad a_2 = -1 + a_1$$

De plus, le domaine des pôles réels est séparé de celui des pôles complexes par la parabole d'équation: $a_1{}^2 = 4.a_2$.

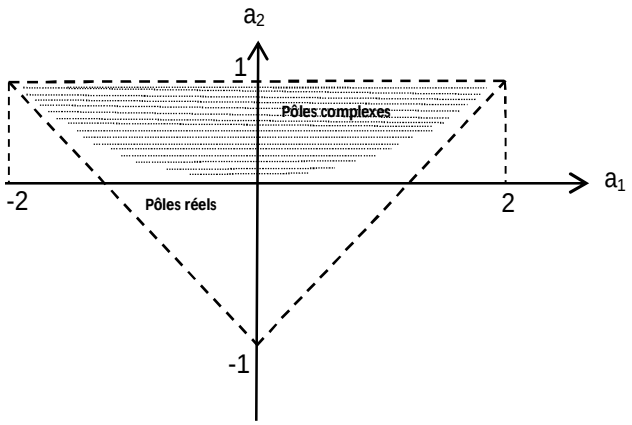


Figure 3.8. Triangle du domaine de stabilité des filtres numériques RII

3.4.3. Synthèse des filtres numériques RII

Les techniques utilisées pour effectuer la synthèse d'un filtre numérique récursif peuvent être classées en trois catégories [46,47]:

- La transposition d'un filtre continu.
- La synthèse directe dans le plan z, par recherche de la position des pôles et des zéros.
- La synthèse à l'aide des méthodes d'optimisation, qui, en général sont des méthodes itératives.

Le résultat de la synthèse est un jeu de coefficients, qui serviront à calculer par une somme pondérée des échantillons du signal d'entrée et du signal de sortie aux instants précédents, la sortie du filtre pour chaque échantillon d'entrée.
Le nombre de coefficients varie en fonction de la sévérité du gabarit des filtres désirés.

La synthèse des filtre numériques, qu'ils soient récursifs (RII) ou non (RIF), nécessite la connaissance d'un certain nombre de paramètres fondamentaux, tels que :

- La fréquence de coupure normalisée ω_p.
- La fréquence de coupure normalisée ω_s.
- La largeur de la zone de transition $\omega_s - \omega_p$.

- Les ondulations autorisées dans la bande passante δ_1.
- Les ondulations autorisées dans la bande coupée δ_1.

La figure ci-dessous, représente un gabarit fréquentiel, d'un filtre passe-bas, illustrant ces paramètres:

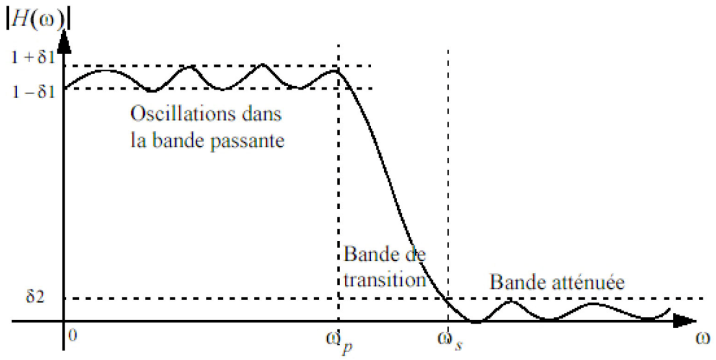


Figure 3.9 : Spécifications fréquentielles (gabarit) d'un filtre passe-bas.

3.5. Propriétés des filtres numériques

Les filtres numériques ont des propriétés plus performantes par rapport aux filtres analogiques. Ces dernières peuvent être résumées en :

- **Souplesse :** La réponse en fréquence peut être très aisément modifiée en changeant les coefficients arithmétiques ; le domaine des fréquences de travail est facilement déplacé par modification de la fréquence d'échantillonnage.
- **Précision :** Les différentes manipulations étant effectuées sur des nombres, la précision ne dépend, en grande partie, que de celle du CAN (convertisseur Analogique-Numérique) et de celle du CNA.
- **Association des filtres :** La mise en série des filtres numériques ne pose aucun problème d'interaction, tel que celui que l'on rencontre pour l'adaptation des impédances des filtres analogiques.
- **Stabilité des caractéristiques :** Il n'y a pas de vieillissement des composants dû à l'influence de la température sur les caractéristiques du filtre.

Cependant, les principaux inconvénients sont liées au problème de l'échantillonnage (spectre du signal toujours limité) nécessitant l'utilisation de processus ayant une bonne rapidité d'exécution pour pouvoir traiter des signaux aux fréquences élevées en temps réel.

3.6. Conclusion

Avec l'importante l'évolution des ordinateurs et des performances des convertisseurs analogique-numérique, qu'on connait actuellement, le filtrage numérique est devenu un outil fondamental dans tous les domaines où l'on traite le signal.

Les méthodes de synthèse des filtres numériques, citées précédemment, permettent de construire des filtres pour des cas simples. Il ne faut pas oublier que le filtrage numérique impose une numérisation préalable du signal issu du capteur, et si les performances exigées sont modestes, cette opération n'est peut-être pas justifiée. Un filtre analogique toujours stable et peu couteux peut suffire. Par contre, un filtre numérique peut avoir des performances hors de portée de tout filtrage analogique, si complexe soit-il. Des techniques sophistiquées sont souvent nécessaires [35], et l'amélioration de techniques numériques de traitement du signal est un sujet de recherche très actuel, et en pleine expansion.

CONCLUSION

Les algorithmes de colonies de fourmis forment une classe de métaheuristiques destinée à résoudre des problèmes d'optimisation difficile, pour lesquels, les méthodes de résolution classiques sont peu efficaces.

Ces algorithmes, qui s'inspirent du comportement des fourmis réelles qui possèdent des caractéristiques collectives et individuelles pouvant contribuer à la résolution de problèmes complexes, sont généralement utilisés comme des méthodes génériques qui peuvent optimiser une large gamme de de différents problèmes, sans nécessiter de changements profonds dans la procédure employée.

D'origine discrète, ces algorithmes ont été élaborés par les informaticiens, pour la résolution des problèmes d'optimisation combinatoire. Le ''Ant System'' , qui est le premier algorithme apparu, a été destiné à la résolution du problème du voyageur de commerce purement combinatoire. Cependant, il existe, en ingénierie, une large gamme de problèmes où la fonction objectif est à variables continues et pour lesquelles, la métaheuristique peut être d'un grand secours (fonction non dérivable, multiples minimums locaux, non convexe, nombre de variables important…). D'où l'apparition de multiples tentatives d'adaptation.

Cette adaptation aux cas continus étant en plein exploitation, et le choix de la recherche locale demeure un facteur très important, afin de produire des algorithmes compétitifs avec d'autres métaheuristiques plus anciennes et souvent plus spécialisées. Ce qui permet de dire, que les algorithmes de colonies de fourmis produits n'ont pas atteint leur pleine maturité.

Dans ce travail de mémoire, nous nous sommes intéressés à l'algorithme de colonies de fourmis **API**, déduit de la modélisation du comportement de fourragement de l'espèce ***Pachycondyla Apicalis***, d'où l'appellation **API,** et son application au problème général de l'optimisation (discret et continu). Les fourmis de cette espèce possèdent en effet la caractéristique de chasser en solitaire et d'utiliser un nombre réduit de règles comportementales pour rechercher leurs proies. Ainsi, à partir de leur nid, ces fourmis couvrent globalement une surface donnée en la

partitionnant en plusieurs sites de chasse individuels. Chaque fourmi effectue une exploration aléatoire locale de sites de chasse et le site choisi est sensible au succès précédemment rencontrés sur les autres sites.

De telles caractéristiques, offrent l'avantage de simplicité de la modélisation, indépendamment de l'espace de recherche considéré, ce qui encourage son utilisation pour d'autres applications.

Cette simplicité de modélisation ne doit pas masquer les difficultés rencontrées lors du réglage des divers paramètres de l'algorithme.

Afin de mettre en œuvre cet algorithme, nous nous sommes proposé, dans ce mémoire de l'appliquer à l'identification des filtres numériques. En effet, les filtres que nous avons tenté d'identifier sont des filtres numériques récursifs au sens de l'approximation de Cauer (filtres elliptiques). Ces filtres présentant plusieurs caractéristiques : des ondulations dans la bande passante et dans la bande coupée, une mince largeur de la bande de transition, une instabilité fréquente et qui est dûe à la nature non récursive des filtres, ont permis de classer notre problème d'optimisation parmi les problèmes d'optimisations complexes, continus et à multi-objectifs.

Cette optimisation a été assurée en s'appuyant sur la minimisation de la fonction objectif, définie par l'inverse d'une fonction d'évaluation qui tient compte des caractéristiques sitées précédemment.

Les résultats obtenus sont compétitifs par rapport à ceux obtenues par d'autres méthodes. Néanmoins, la procédure employée (API) présente l'avantage de rapidité de convergence, qui est un facteur très important.

Les perspectives de ce travail sont relativement nombreuses, citons :

- La synthèse des filtres numériques en précision finie.
- L'hybridation de l'API avec d'autres heuristiques d'optimisation, dans le but, d'améliorer la convergence de l'algorithme. On pourra, par exemple, changer la règle de décision d'une fourmi lors de la capture d'une proie en utilisant une heuristique secondaire.

- L’utilisation de « modèle multi populations », qui permettra à l’algorithme de démarrer en différents points de l’espace de recherche S. La stratégie consistera à utiliser plusieurs colonies de fourmis, chacune avec un nid positionné initialement à des positions différentes dans S.

REFERENCES

1. M.Dorigo, T.Stűzle, ″Handbook of Metaheuristics″, tome 57 de International series in operations research and management science, chapitre ″The Ant Colony Optimization Metaheuristics: Algorithms, Applications and Advances″. Kluwer Academic Publishers, Boston Hardbound, 2003.

2. J.Dréo, A. Pétrowski, E.Taillard, Patrick Siarry, ″Métaheuristiques pour l'optimisation difficile″, éditions Eyrolles, 2003.

3. M. Hadj Sadok, ″API une métaheuristique pour l'optimisation difficile. Application à l'identification de la machine asynchrone″, thèse de doctorat d'état en électronique, Université S.Dahleb, Blida, 2007.

4. A.Berro, ″Optimisation multi objectif et stratégie d'évolution en environnement dynamique″, thèse PhD, Université Paul Sabatier, Toulouse, 2001.

5. E.Talbi, ″A taxonomy of hybrid metaheuristics″, Journal of heuristics, tome 8, 2002, 541-564.

6. C.Andréa, L.Thé Van, M.Guillaume, ″Optimisation par colonies de fourmis″, Mai 2006.

7. J.Dréo, P.Siarry, ″Algorithmes à estimation de distribution et colonies de fourmis.″, 11ème journée évolutionnaire (JETII).

8. M.Dorigo, G.Dicaro,″The Ant Colony optimization meta-heuristic″,In Corne, D, M.Dorrigo, F.Glover, (Eds), News ideas in optimization, McGraw-Hill, London, 1999, 11-32.

9. J.L.,Deneubourg, S.Goss, N.Franks, C.Detrain, L.Chretien, ″The dynamics of collecting storing:robot-like and ant-like robots″, in Meyer and Wilson, 1990, 356-365.

10. E.Taillard, ″FANT: Fast Ant System″, Technical Report 46-98, IDSIA, Lugano, 1998.

11. J.DREO, ″Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical″, thèse de doctorat, en Sciences de l'ingénieur, Université Paris 12, VAL DE MARNE, Décembre 2004.

12. M.Dorigo, V.Maniezzo, A.Colorni, ″The Ant System: optimization by a colony of cooperating agents″, IEEE Irons systems.

13. J.W.Gutjahr, ″A graph-based Ant System and its convergence″, Future generation computing systems, tome 16, No 8,873-888, 2000.

14. J.W.Gutjahr, ″ACO algorithms with guarantied convergence to the optimal solution″, Information Processing Letters, tome 82, No 3, 987-999,2002.

15. A.Colorni, M.Dorigo, V.Maniezzo, ″New Results of an Ant System Approch applied to the assymetric TSP″, LH.Osman and Kelly,J editors, Metaheuristics International conference, Hilton Breckenridge, Colorado. Kluwer academic publishers.

16. V.Maniezzo, A.Colorni, M.Dorigo, ″The Ant System applied to the quadratic assignment problem″, Technical report 94-28, IRIDIA, université Libre de Bruxelles, Belgium, 1994.

17. L.M.Gambardella, E.D.Taillard, M.Dorigo, ″Ant colonies for the quadratic assignment problem″, journal of the operational research society, vol 50, No 02, 167-176, 1999.

18. C, H.Hooks, ″Max-Min ant system″, Future Generation Computing Systems, vol 16, No 8, 889-914, June 2000.

19. D.Costa, A.Hertz, ″Ants can color graphs″, Journal of the operational research society, Vol 48, No 3, 295-305, Mars 1997.

20. V.Maniezzo, A.Carbonaro, ″An Ants Heuristic for the frequency assignment problem″, future generation computing systems, 16(8), 927-935, 2000.

21. G.Bilchev, I.C.Parmee, ″The ant colony metaphor for searching continuous design spaces″, In Fogarty, T.C.(Ed), Proc. of the AISB Workshop on Evolut, Computa, Vol 993 of LNCS, Springer- Verlag, Berlin, Germany, 25-39, 1995.

22. K.Socha, ″ACO for continuous mixed-variable optimization″, In M.Dorigo, M.Birattari, C.Blum, L.M.Gambardella, F.Mondada and T.Stűzle editors,, Ant Colony Optimization and Swarm Intelligence, Vol 3172 of lecture notes in Computing Sciences, Springer 2004, 25-36.

23. E.D.Lumer, B.Faieta, ″Diversity and Adaptation in Populations of Clustering ants″, Proceedings of the Third International Conference on simulation of Adaptative Behaviour, 501-508, 1994.

24. J.Dréo, P.Siarry, ″Un algorithme de colonies de fourmis en variables continues hybridé avec un algorithme de recherché locale″, 5ème congrès de la société Française de recherche opérationnelle et Aide à la décision (ROADEF 2003).

25. M.Mathur, S.B.Karale, S.priye, V.K.Jyaraman, B.D.Kulkarni, ″ Ant Colony approach to continuous function optimization″, Industrial and Engineering Chemistry Research, Vol 39, 3814-3822, 2000.

26. C.Ling, S.Jie, Q.Ling, C.Hongjian, ″A method for solving optimization problems in continuous space using ant colony algorithm″, Proc. Of the Third Internat. Workshop on Ant Algorithms (Ants'2002), lecture notes in computer science, 288-289, 2002.

27. J.Dréo, P.Siarry, ″Continuous interacting ant colony algorithm based on dense hierarchy″, Future Generation Computer Systems, Vol 20, No 5, 841-856, 2004.

28. .Dréo, P.Siarry, ″Un nouvel algorithme de colonies de fourmis exploitant la communication directe entre individus, application à l'optimisation en variables continues″, 4ème congrès de la société Française de recherche opérationnelle et Aide à la décision (ROADEF 2002).

29. E.Wilson, B.Holldobler, ″Dense Heterarchy and mass communication as the basis of organization in ant colonies″, Trend in Ecology and Evolution, tome 3, 65-68, 1988.

30. G.Venturini, M.Slimane, N.Monmarché, D.Fresneau, ″Modélisation des fourmis *Pachycondyla Apicalis* appliquée à l'optimisation numérique″, rapport interne 194, Laboratoire d'informatique de l'université de Tours, E3I Tours, 1997.

31. P.Jaisson, ″La fourmi et le sociobiologiste″, Odile Jacob, Paris, 1993.

32. L.M.Gambardella, M.Dorigo, ″Solving symetric and assymetric TSPs by ant colonies″, Proc. IEEE Conf. on Evolut Computa. ICEC,New Jersy , 622-627, 1996.

33. N.Monmarché, ″Algorithmes de fourmis artificielles: application à la classification et à l'optimisation″, thèse de doctorat, Laboratoire d'informatique, Université de Tours, France, 2000.

34. M.Dorigo, L.M.Gambardella, ″Ant colony system : A cooperative learning approch to the Travelling Salesman Problem″, IEEE Trans, on Evolutionary Computation, Vol 1, No 1, 53-66, April 1997.

35. W. TFAILI, ″Conception d'un algorithme de colonies de fourmis pour l'optimisation continue dynamique″, thèse de doctorat en sciences de l'ingénieur, Université Paris 12, Val De Marne, Décembre 2007.

36. P.Siarry, ″Optimisation en traitement du signal et de l'image″, éditions Hermès, Lavoisier, 2007.

37. A.Antoniou. ″Digital Filters: Analysis, Design and Applications″. McGraw-Hill Companies; 2nd edition (January 1, 1993).

38. F.Cottet. ″Aide Mémoire : Traitement du signal″. Edition Dunod, Paris 2000.

39. Monson H.Hayes, ″Schaum's outline of theory and problems of digital signal processing″. Schaum's outline series, Mc Graw-Hill © 1999.

40. M.Bellanger, ″Traitement du signal : Théorie et Pratique″, 8ème édition.

41. L.R. Rabiner and B. Gold, ″Theory and Application of Digital Signal Processing″,
New Jersey: Prentice-Hall, Inc,1975.

42. A.V. Oppenheim and R. W. Schafer, ″Digital Signal Processing″, Englewood Cli_s,
Prentice-Hall, Inc,1975.

43. D.W. Tufts and J.T. Francis, ″ _Designing digital lowpass filters: Comparison of
some methods and criteeria″, IEEE Trans. Audio Electroucoust., vol. AU-18, pp.
487-494, December 1970.

44. W. C. Kellogg, ″Time domain design of nonrecursive least mean square digital filters″, IEEE Trans. Audio Electroacoust., Vol. AU-20, No. 2, pp. 155-158, June 1972.

45. Y.C. Lim and S.R. Parker, ″Discrete coeffcient FIR digital filter design based upon an LMS criteria″, IEEE Trans. Circuits Syst., Vol. CAS-30, No. 10, pp. 723-739, October 1983.

46. E.Ya. Remez, ″Sur la détermination des polynômes d'approximation de degré donnée″, Comm. Soc. Math. Kharkov, Vol. 10, pp. 41-63, 1934.

47. T.W. Parks and J.H. McClellan, ″Chebyshev approximation for nonrecursive digital filters with linear phase″, IEEE Trans. Circuit Theory, vol. CT-19, No. 2,pp. 189_194, Mach 1972.

48. J.Prado, ″Filtres numériques: Synthèse″, notes de cours, Ecole nationale supérieure des télécommunications de Paris.

49. G.Wade, ″Signal coding and processing″, 2ème edition, 1994.

50. Brian D.Hann, Daniel T.Valentine, ″Essential MATLAB ® for Engineers and Scientists″, Liance house, Jordan Hill, Oxford, third edition 2007.

51. M. Frikel, G.Binet, ″Filtres à réponse impulsionnelle finie″, notes de cours, 1ère édition, copyright 2000, Ecole nationale supérieure d'ingénierie de Caen.

52. R.Ben Atitallah, A.Ben Hamida, N.Masmoudi, ″Optimisation de filtres numériques en vue d'une conception d'une stratégie de simulation pour la prothèse cochléaire″, Laboratoire d'électronique et des technologies de l'information, Ecole nationale d'Ingénieurs de Sax.