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

Université Saad Dahlab, Blida
USDB

Faculté des Sciences
Département Informatique

Mémoire pour l'obtention
d'un diplôme d'Ingénieur d'Etat en informatique

Option : Intelligence Artificielle

Sujet :

Optimisation d'un clavier visuel *à l'aide d'un algorithme de* colonies de fourmis

Présenté par : Alikacem Larbi Amine
Chaib Fayçal

Promoteur : Dr. Oukid–khaouas Saliha
Encadreeur : Chabane Yahia

Organismes d'accueil : LRDSI, Université de Blida.

Président : Souami
Examinateur : Mr Hadj Sadouk
Examinateur : Mme Benblidia

2007- 2008

Dédicaces

Je dédie ce travail,

A mes chers parents, mon père, ma mère pour leur amour, leurs encouragements et leurs sacrifices.

A mes très chers frères et sœurs,

A mes neveux, mes nièces,

Ainsi qu'à toute ma famille

A mes amis : Mohamed, Walid, Yacine, Nabih, Youcef, Reda, Mouloud, Riadh et Khaled.

A mon binôme Larbi et à toute sa famille.

Je tiens spécialement à mentionner le plaisir que j'ai eu à travailler avec Mr. Chabane yahia et Mr. Chabane Djamel dont les conseils, orientations et suggestions ont étaient plus que déterminants dans l'aboutissement de ce travail en les remerciant pour toute leur patience et leur encouragements.

A tous mes amis (es) et collègues qui m'ont accompagné et soutenu durant ces années d'étude.

A tous ceux et celles dont les noms ne sont pas cités je dédie ce travail.

Fayçal Chaib

Dédicace

Je dédie ce travail,

A mes parents qui m'ont toujours poussés et motivés dans mes études, à mes frères et sœur ce mémoire représente donc l'aboutissement du soutien et des encouragements qu'ils m'ont prodigués tout au long de ma scolarité. Qu'ils en soient remerciés.

A tous les membres des familles Alikacem et damerdji

A tous mes amis de l'université de Blida avec lesquels j'ai eu le plaisir de partager mon cursus universitaire, pour leur soutien moral et pour la contribution technique qu'ils m'ont apporté.

Je remercie Mme Oukid-khaouas pour son encadrement technique qui a porté une aide décisive à l'aboutissement de ce travail.

Je tiens spécialement à mentionner le plaisir que j'ai eu à travailler avec Mr Chabane yahia et Chabane Djamel dont les conseils, orientations et suggestions ont étaient plus que déterminants dans l'aboutissement de ce travail en les remerciant pour toute leur patience à notre égard.

Je pense également à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Alikacem Larbi Amine

Résumé

Aujourd'hui, de nombreuses personnes ayant des facultés motrices très réduites dépendent fortement de leur ordinateur pour travailler ou communiquer. Cependant, pour ces personnes, l'utilisation du clavier de l'ordinateur peut être source de fatigue.

L'objectif de notre travail est de réaliser à l'aide des algorithmes de colonie de fourmis un système qui permet d'organiser la disposition des touches d'un clavier visuel par rapport à un texte donné, pour un but de minimisation de l'effort de saisie (cruciale chez des personnes a mobilité réduite).

Mots clés: Optimisation combinatoire; Algorithmes de colonies de fourmis.

Abstract

Today, many people with reduced moving faculty, highly depends of their computer to work or communicate. However, for these persons using computer keyboard might be tiredness.

The aim of our work is to use ant colony algorithm in order to solve the problem of the arrangement of letters on a visual computer keyboard using a given text, to minimize typing effort (crucial for reduced mobility persons).

Keywords: Ant Colony Algorithm; Keyboard arrangement.

SOMMAIRE

Introduction Générale

L'utilisateur d'un clavier d'ordinateur est souvent surprit dès les premières confrontations de la disposition pseudo-aléatoire des touches, les dispositions les plus courantes sont « l'AZERTY » pour les claviers à vocation francophone et le « QWERTY » pour les claviers à usage anglophone, ces dispositions ont été mises en place sur les machines à écrire au début du 19ème siècle et qui permettent de réduire le blocage de ces dernières.

Aujourd'hui, de nombreuses personnes ayants des facultés motrices très réduites dépendent fortement de leur ordinateur pour travailler ou communiquer. Cependant, pour ces personnes l'utilisation du clavier de l'ordinateur peut s'avérer difficile. Pour aider ces personnes à saisir du texte, de nombreux systèmes existent, ils se répartissent en trois (03) principales catégories :

1) modification du matériel : clavier avec des touches plus grandes (clavier NAPIC1), clavier ergonomique.

2) logiciel de simulation de clavier : par émulation d'un clavier à l'écran (clavier virtuel de Windows).

3) logiciel de simulation de clavier qui améliore la vitesse de saisie des utilisateurs par un système de prédiction.

Dans notre étude, nous nous intéressons aux types de systèmes de la deuxième catégorie plus particulièrement.

Les recherches sur les comportements collectifs des insectes sociaux fournissent aux informaticiens des méthodes puissantes pour la conception d'algorithmes d'optimisation. Le comportement des insectes sociaux est caractérisé par l'auto-organisation. Les individus communiquent en changeant les propriétés locales de leur environnement, et par le biais de ce moyen de communication limité, une sorte d'intelligence collective émerge. Les fourmis naturelles ont inspiré les

"algorithmes de colonies de fourmis" introduits par la thèse de Marco Dorigo. L'idée générale est d'imiter le comportement coopératif d'une colonie de fourmis pour résoudre des problèmes complexes d'optimisation combinatoire. Le principe est le suivant :

Les fourmis cherchent de la nourriture et se déplacent de façon quasi aléatoire. Tout au long de leur déplacement elles laissent derrière elles une substance chimique appelée "phéromone". Cette substance a la propriété de s'évaporer au cours du temps et a pour but de guider les fourmis vers leur objectif. Une fois cet objectif atteint (dans notre cas, la nourriture trouvée), les fourmis rentrent au nid en empruntant le même chemin qu'à l'aller, grâce à leur trace de phéromone. Celle-ci s'en trouve renforcée. Plus une trace de phéromone est concentrée, plus elle va attirer les fourmis. Au fil du temps, on va donc constater l'émergence du plus court chemin vers la nourriture grâce au renforcement de la trace de phéromone.

L'objectif de notre travail est de réaliser à l'aide des algorithmes de colonie de fourmis un système qui permet d'organiser la disposition des touches d'un clavier visuel par rapport à un type de texte donné, pour le but de minimiser l'effort de frappe (cruciale chez des personnes a mobilité réduite).

Notre travail va se repartir en trois étapes :

1. Etude des méthodes d'optimisation et des algorithmes de colonie de fourmis.
2. Conception du système.
3. Test du système et évaluation des résultats.

Chapitre I

LES METHODES D'OPTIMISATION

Chapitre I : LES METHODES D'OPTIMISATION

I.1 Introduction

Dans les vingt dernières années, on a vu que l'ensemble des techniques mathématiques et algorithmiques de résolution de problèmes d'optimisation se développent considérablement. Les progrès significatifs des techniques d'évaluation associés à l'augmentation considérable de la capacité de calcul des machines permettent aujourd'hui de traiter des problèmes de plus en plus complexes, avec des tailles des données de plus en plus importantes. L'une des conséquences de ceci est que la construction même des modèles, de façon fiable et efficace, n'est plus un problème secondaire mais elle est devenue un problème central.

En outre, d'autres problèmes apparaissent, aussi bien dans le monde des techniques numériques que dans les méthodes de simulation. Du point de vue des premières, la résolution numérique de systèmes d'équations de grande taille (millions ou milliards d'équations et d'inconnues, voire une infinité) n'est pas une tâche simple. En ce qui concerne la simulation, la prise en charge de la rareté de certains événements est un problème non trivial.

L'optimisation est un sujet central en recherche opérationnelle. Un grand nombre de problèmes d'aide à la décision pouvant en effet être décrits sous la forme de problèmes d'optimisation. Parmi les problèmes rencontrés par le chercheur et l'ingénieur "l'optimisation " ou " programmation mathématique " (Mathematical Programming) sont des termes utilisés pour recouvrir toutes les méthodes qui servent à déterminer l'optimum d'une fonction avec (ou sans) contraintes. Citons par exemple les problèmes d'identification, l'apprentissage supervisé de réseaux de neurones, la recherche du plus court chemin, la conception de systèmes mécaniques ou encore le traitement des images.

Ce chapitre décrit tout d'abord le cadre de l'optimisation difficile et des métaheuristiques dans lequel nous nous plaçons dans ce travail, nous allons définir les types d'optimisations ainsi que les différentes méthodes de résolution des problèmes d'optimisation.

I.2 Problème d'optimisation

Un problème d'optimisation au sens général est défini par un ensemble de solutions possibles ***s***, dont la qualité peut être décrite par une fonction objectif ***f***, On cherche alors à trouver la solution s^* possédant la meilleure qualité $f(s^*)$ (par la suite, on peut chercher à minimiser ou à maximiser $f(s)$). Un problème d'optimisation peut présenter des contraintes d'égalité (ou d'inégalité) sur s, être dynamique si $f(s)$ change avec le temps ou encore multi-objectif si plusieurs fonctions objectives doivent être optimisées.

Il existe des méthodes déterministes (dites "exactes") permettant de résoudre certains problèmes en un temps fini. Ces méthodes nécessitent généralement un certain nombre de caractéristiques de la fonction objective, comme la stricte convexité, la continuité ou encore la dérivabilité. On peut citer comme exemple de méthode : la programmation linéaire, quadratique ou dynamique, la méthode du gradient, la méthode de Newton, etc.

La résolution des problèmes d'optimisation est utilisée dans un grand nombre de domaines [AARTS & LENSTRA, 1997]. A l'origine, ce sont les militaires qui se sont intéressés à ces questions au cours de la seconde guerre mondiale. C'était en fait un nouveau domaine de recherche en mathématiques appliquées qui a vu le jour avec la recherche opérationnelle. Le développement de l'informatique a ouvert de nouveaux horizons à la résolution de ces problèmes, et a permis un élargissement massif des champs d'application de ces techniques.

La résolution d'un problème d'optimisation et un problème complexe, car de nombreux facteurs interviennent et interagissent entre eux. Néanmoins, l'optimisation appliquée au domaine informatique et de la recherche opérationnelle permet de résoudre des problèmes qui étaient insolubles auparavant et aboutit souvent à des solutions originales.

Optimisation difficile

Certains problèmes d'optimisation demeurent cependant hors de portée des méthodes exactes. Un certain nombre de caractéristiques peuvent en effet être problématiques, comme l'absence de convexité stricte (multi modalité), l'existence de discontinuités, une fonction non dérivable, présence de bruit, etc.

Dans de tels cas, le problème d'optimisation est dit "difficile", car aucune méthode exacte n'est capable de le résoudre exactement en un temps "raisonnable", on devra alors faire appel à des heuristiques permettant une optimisation approchée.

L'optimisation difficile peut se découper en deux types de problèmes :

- les problèmes discrets
- les problèmes continus

Le premier cas rassemble les problèmes de type NP-complets, tels que le problème du voyageur de commerce.

Un problème "NP" est dit complet s'il est possible de le décrire à l'aide d'un algorithme polynomial sous la forme d'un sous-ensemble d'instances. Concrètement, il est "facile" de décrire une solution à un tel problème, mais le nombre de solutions nécessaires à la résolution croît de manière exponentielle avec la taille de l'instance. Jusqu'à présent, l'hypothèse postulant que les problèmes NP-complets ne sont pas solubles en un temps polynomial n'a été ni démontrée, ni révoquée. Aucun algorithme polynomial de résolution n'a cependant été trouvé pour de tels problèmes. L'utilisation d'algorithmes d'optimisation permettant de trouver une solution approchée en un temps raisonnable est donc courante.

Dans la seconde catégorie, les variables du problème d'optimisation sont continues. C'est le cas par exemple des problèmes d'identifications, où l'on cherche à minimiser l'erreur entre le modèle d'un système et des observations expérimentales. Ce type de problème est moins formalisé que le précédent, mais un certains nombre de difficultés est bien connues, comme l'existence de nombreuses variables présentant des corrélations non identifiées, la présence de bruit ou plus généralement une fonction objectif accessible par simulation uniquement. En pratique, certains problèmes sont mixtes et présente à la fois des variables discrètes et des variables continues.

L'optimisation difficile pose deux sortes de problèmes. Ceux-ci reçoivent dans la littérature cette appellation non définie strictement (liée en fait à l'état de l'art en matière d'optimisation). Certains problèmes particuliers d'optimisation discrète, pour lesquels on ne connaît pas d'algorithme exact *polynomial* (c'est-à-dire dont le temps de calcul est proportionnel à N^n, où N désigne le nombre de paramètres inconnus du problème, et n est une constante entière). C'est le cas, en particulier, des problèmes dits « NP-difficiles», pour lesquels on hypothèse qu'il n'existe pas de constante n telle que le temps de résolution soit borné par un polynôme de degré n et certains problèmes d'optimisation à variables continues, pour lesquels on ne connaît pas d'algorithme qui permet de repérer un *optimum global* (c'est à dire la meilleure solution possible) à cout sûr et en un nombre fini de calculs.

Des efforts de recherche ont longtemps été menés, séparément, pour résoudre ces deux types de problèmes. Dans le domaine de l'optimisation continue, il existe ainsi un arsenal important de méthodes classiques dites d'*optimisation* [Berthiau & Siarry, 2001]. Mais ces techniques sont souvent inefficaces si la fonction objectif ne possède pas une propriété structurelle particulière, telle que la convexité. Dans le domaine de l'optimisation discrète, un grand nombre d'*heuristiques* qui produisent des solutions proches de l'optimum, ont été développées. Mais la plupart d'entre elles ont été conçues spécifiquement pour un problème donné.

I.3 Les éléments d'optimisation

L'optimisation est une méthode mathématiques consacré à l'étude du (ou des) minimum(s)/maximum(s) d'une fonction à une ou plusieurs variables sur un certain domaine de définition, de l'étude de leur existence à leur détermination. En général elle passe par la mise en œuvre d'un algorithme et par la suite un programme. Pour mener à bien une opération, plusieurs éléments sont indispensables et conditionnent la solution trouvée. La figure suivante présente les quatre éléments essentiels à la résolution d'un problème d'optimisation.

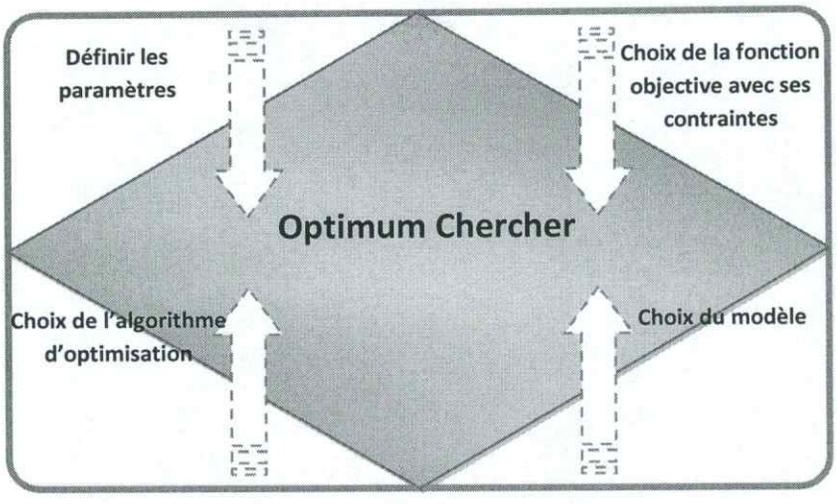


Figure I.1 Eléments indispensables d'optimisation.

En général, un grand nombre de paramètres sont indispensables, il faut être capable de définir les paramètres utiles à l'optimisation. Certains paramètres ont une influence sur la fonction choisie. Etant donné le coût des simulations, seul les paramètres influents sont à retenir :

Une fonction objective : définie l'objectif à atteindre. La définition de cette fonction est en fait un problème délicat. Car le problème est formulé en un problème d'optimisation par l'intermédiaire de la fonction objective. C'est elle qui est au centre de l'optimisation.

Un modèle : précis, robuste et malléable. Ce modèle doit être utilisable sur un domaine d'étude le plus large possible.

Un algorithme d'optimisation : permet de trouver la solution. Différentes méthodes d'optimisation existent.

I.4 Optimisation combinatoire

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre

part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace.

Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (RO) et en intelligence artificielle (IA).

I.5 Classification des méthodes de résolution

I.5.1 Les méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles (développées depuis une trentaine d'années) telles que les techniques de séparation et l'évaluation progressive (SEP) ou les algorithmes avec retour arrière. Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable.

Malgré les progrès réalisés(notamment en matière de la programmation linéaire en nombres entiers), comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de taille importante.

I.5.2 Méthodes approchées

Les méthodes approchées fournissent une solution approchée au problème traité, elles sont en général conçues de manière à ce que la solution obtenue puisse être située au voisinage de la valeur optimale : de telle méthodes permettent d'obtenir des bornes inférieures ou supérieures de la valeur optimale tel que : Les Méthodes Méta heuristiques.

I.5.2.1 La démarche heuristique

Heuristique (du grec heuriskêin, « trouver ») est un terme qui signifie l'art d'inventer, de faire des découvertes. C'est en sociologie, une discipline qui se propose de dégager les règles de la recherche scientifique [Larousse].

En optimisation combinatoire, Théorie des graphes et Théorie de la complexité, une heuristique est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation NP-difficile. Une heuristique, ou méthode approximative, est donc le contraire d'un algorithme exact qui trouve une solution optimale pour un problème donné. Les algorithmes de résolution exacts étant de complexité exponentielle, il est généralement plus judicieux de faire appel à des méthodes heuristiques pour des problèmes difficiles.

L'usage d'une heuristique est approprié pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution. Généralement, une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre, mais les approches peuvent contenir des principes plus généraux. On parle de *métaheuristique*, pouvant s'appliquer à différents problèmes (comme le recuit simulé par exemple).

La qualité d'une heuristique peut s'évaluer selon deux critères scientifiques :

1) *Critère pratique :* on implémente l'algorithme approximatif et on évalue la qualité de ses solutions par rapport aux solutions optimales (ou aux meilleures solutions connues). Ceci passe par la mise en place d'un ensemble d'instances d'un même problème accessible à tous.

2) *Critère mathématique :* il faut démontrer que l'heuristique garantit des performances. La garantie la plus solide est celle des algorithmes approchés, sinon il est intéressant de démontrer une garantie probabiliste, lorsque l'heuristique fournit souvent, mais pas toujours, de bonnes solutions.

L'heuristique [Roux, 2001] est une méthode, une technique ou un critère de guidage ou de décision, en général, expérimental ou obtenu par approximation, permettant de choisir la voie la plus prometteuse de recherche de la solution au problème posé, ou d'éliminer les voies les moins intéressantes, sans garantie sur la validité ou la précision de l'information ainsi fournie.

Entrer dans le domaine des heuristiques, c'est se départir d'emblée les schémas classiques. En effet, alors que la démarche classique mathématique est centrée sur l'objet de l'étude, sur la compréhension de sa structure et de sa logique, la démarche heuristique repousse le problème lui-même au rang d'illustration pour dégager des schémas de pensée plus généraux et donc originaux.

Les heuristiques disposent d'une simplicité et donc d'une rapidité dans leur exécution plus élevée que les algorithmes classiques. Ces règles s'appliquant à un ensemble particulier, la recherche des faits ce voit simplifiée et accélérée (moins de possibilité), d'où une analyse des situations améliorées. Mais une méthode heuristique trop simplifiée ou au contraire trop générale peut conduire à des erreurs de décision.

L'utilisation de plus de ces éléments simples (les heuristiques) afin de créer des éléments plus complexes (les méta-heuristiques) permet donc de réduire considérablement l'ensemble de recherche global de l'algorithme.

L'une de leur caractéristique principale et à première vue défaut, dont hérite également les méta-heuristiques, est qu'ils peuvent dans certains cas ne pas proposer de solution optimale au problème. Mais au résultat s'y approchant d'assez près pour qu'il soit considéré comme correct, on parle alors de garantie de performance.

Une heuristique d'optimisation est une méthode approchée se voulant êtres simple, rapide et adaptée à un problème donné. Sa capacité à optimiser un problème avec

un minimum d'informations est contrebalancée par le fait qu'elle n'offre aucune garantie quant à l'optimalité de la meilleure solution trouvée.

Du point de vue de la recherche opérationnelle, ce défaut n'est pas toujours un problème, tout spécialement quand seule une approximation de la solution optimale est recherchée.

I.5.2.2 Métaheuristiques

Les métaheuristiques sont souvent employées pour leur facilité de programmation et de manipulation.

Elles sont en effet facilement adaptables à tout type de problème d'optimisation.

Toutefois, elles sont le plus judicieusement employées sur des problèmes d'optimisation difficile, où des méthodes d'optimisation plus classiques (méthodes déterministes, notamment) montrent leurs limites.

De façon générale, on peut considérer que des problèmes présentant les caractéristiques suivantes sont assez propices à l'utilisation de métaheuristiques :

- NP-complétude,
- nombreux optima locaux,
- discontinuités,
- contraintes fortes,
- non dérivabilité,
- temps de calcul de la fonction objectif excessif,
- solution approchée souhaitée,
- etc.

Les méta-heuristiques sont apparues dans les années 1980 et forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile, pour lesquels on ne connaît pas de méthode classique plus efficace. Elles sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé [Bullnheimer & Hartl & Strauss, 1999] [Corne & Dorigo & Glover, 1999].

La recherche d'une solution optimale pour un problème d'optimisation combinatoire est souvent difficile dans les applications pratiques en raison de la dimension des problèmes. Les méthodes exactes rencontrent généralement des difficultés face aux applications de taille importante en raison du cout en temps combinatoire, d'où la nécessité de développer des heuristiques pour remédier aux problèmes de cout.

Une heuristique est une procédure qui exploite au mieux la structure du problème à optimiser dans le but de trouver une solution raisonnable (non nécessairement optimale) en un temps réduit. Les performances d'une heuristique sont liées à la qualité de la solution produite ainsi qu'au temps de calcul nécessaire pour son obtention.

Parmi les heuristiques, certaines sont adaptables à un grand nombre de problèmes différents sans changements majeurs dans l'algorithme, on parle alors de méta-heuristiques. La plupart des heuristiques et des métaheuristiques utilisent des processus aléatoires comme moyens de récolter de l'information et de faire face à des problèmes comme l'explosion combinatoire. En plus de cette base stochastique, les métaheuristiques sont généralement itératives, c'est-à-dire qu'un même schéma de recherche est appliqué plusieurs fois au cours de l'optimisation, et directes, c'est-à-dire qu'elles n'utilisent pas l'information du gradient de la fonction objectif. Elles tirent en particulier leur intérêt de leur capacité à éviter les optima locaux, soit en acceptant une dégradation de la fonction objectif au cours de leur progression, soit en utilisant une population de points comme méthode de recherche (se démarquant ainsi des heuristiques de descente locale).

La décomposition (méta-heuristique) : méta qui signifie « au delà » ou « plus haut » en grec et de « heuristique » qui signifie « trouver » permet de facilement comprendre le but premier de ces algorithmes : trouver des solutions à des problèmes en utilisant plusieurs (méta) heuristiques.

Souvent inspirées d'analogies avec la réalité (physique, biologie, éthologie, . . .), elles sont généralement conçues au départ pour des problèmes discrets, mais peuvent faire l'objet d'adaptations pour des problèmes continus.

Les métaheuristiques, du fait de leur capacité à être utilisées sur un grand nombre de problèmes différents, se prêtent facilement à des extensions. Pour illustrer cette caractéristique, citons notamment :

— l'optimisation multi-objectif (dites aussi multicritère) [Collette and Siarry, 2002], où il faut optimiser plusieurs objectifs contradictoires. La recherche vise alors non pas à trouver un optimum global, mais un ensemble d'optima formant la "surface de compromis" du problème.

— l'optimisation multimodale, où l'on cherche un ensemble des meilleurs optima globaux et/ou locaux.

— l'optimisation de problèmes bruités, où il existe une incertitude sur le calcul de la fonction objectif. Incertitude dont il faut alors tenir comptes dans la recherche de l'optimum.

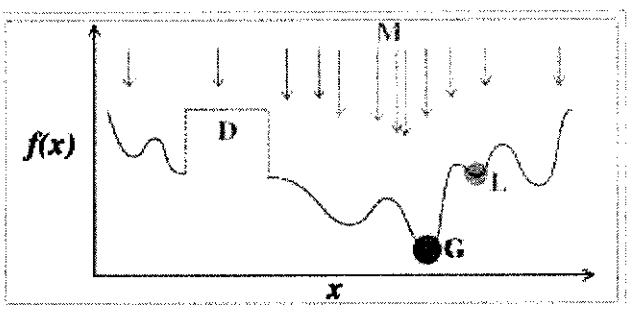


Figure I.2: Les métaheuristiques (M) sont souvent des algorithmes utilisant un échantillonnage probabiliste. Elles tentent de trouver l'optimum globale (G) d'un problème d'optimisation difficile (avec des discontinuités ---D---, par exemple), sans être piégé par les optima locaux (L).

— l'optimisation dynamique, où la fonction objective varie dans le temps. Il faut alors approcher au mieux l'optimum à chaque pas de temps.

— la parallélisassions, où l'on cherche à accélérer la vitesse de l'optimisation en répartissant la charge de calcul sur des unités fonctionnant de concert. Le problème revient alors à adapter les métaheuristiques pour qu'elles soient distribuées.

— l'hybridation, qui vise à tirer parti des avantages respectifs de métaheuristiques différentes en les combinant [Talbi, 2004].

Enfin, la grande vitalité de ce domaine de recherche ne doit pas faire oublier qu'un des intérêts majeurs des métaheuristiques est leur facilité d'utilisation dans des problèmes concrets. L'utilisateur est généralement demandeur de méthodes efficaces permettant d'atteindre un optimum avec une précision acceptable dans un temps raisonnable. Un des enjeux de la conception des métaheuristiques est donc de faciliter le choix d'une méthode et de simplifier son réglage pour l'adapter à un problème donné.

Donc, une *méta-heuristique est une méthode approchée générique dont le principe de fonctionnement repose sur des mécanismes généraux indépendants de tous les problèmes.*

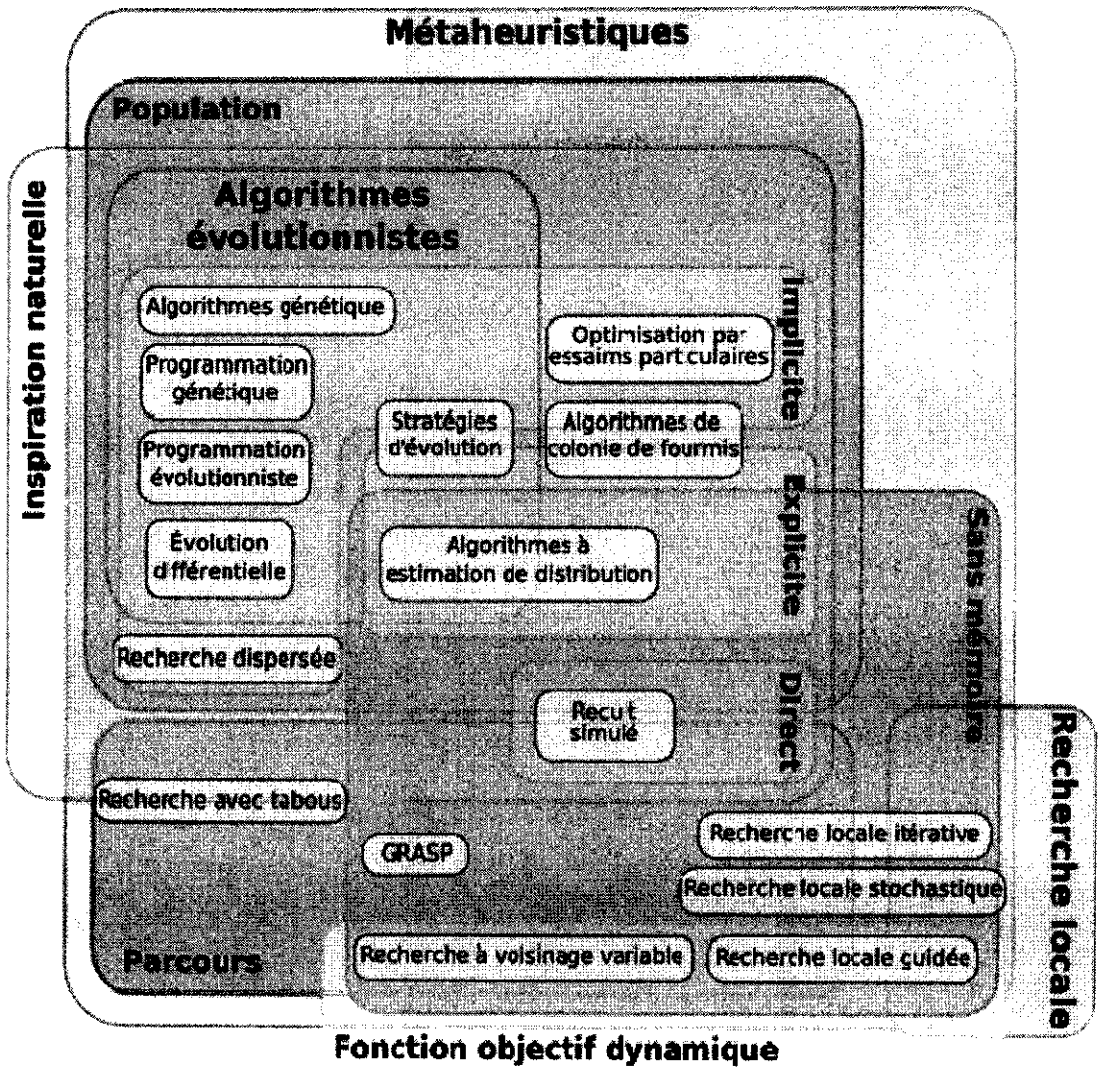


Figure I.3 : Les métaheuristiques peuvent être classées de nombreuses façons. Ce diagramme tente de présenter où se placent quelque unes des méthodes les plus connues. Un élément présenté à cheval sur différentes catégories indique que l'algorithme peut être placé dans l'une ou l'autre classe, selon le point de vue adopté.

Les méta-heuristiques sont souvent inspirées par des systèmes naturels, qu'ils soient pris en physique (les méthodes de voisinage comme le recuit simulé et la recherche tabou), en biologie de l'évolution (les algorithmes évolutifs comme les algorithmes génétiques et les stratégies d'évolution) ou encore en étiologie (les algorithmes de colonies de fourmis). Dans ce qui suit, on présente le principe de certaines méthodes.

I.5.2.2.1 Exemple de métaheuristiques :

I.5.2.2.1.1 Méthodes des algorithmes génétiques :

a) Origines

Les algorithmes génétiques permettent d'obtenir une solution approchée, en un temps correct, à un problème d'optimisation, ils utilisent la notion de sélection naturelle développée par le scientifique Charles Darwin au XIXème siècle.
Dans cette théorie, une population d'individus évolue grâce au mécanisme de la reproduction sexuée. Les individus les plus adaptés à leur milieu se reproduisent plus que les autres, favorisant les caractères les plus adaptés. Ainsi une girafe avec un cou plus long que les autres aura accès à plus de nourriture, et aura donc plus de chances de survivre et de se reproduire. Ses descendants auront un cou plus long, et en moyenne la population de girafe aura un cou plus long.

L'utilisation d'algorithmes génétiques dans la résolution de problèmes est à l'origine des recherches de John Holland dès 1960. La nouveauté introduite a été la prise en compte de l'opérateur crossing over en complément des mutations, et c'est cet opérateur qui permet le plus souvent de se rapprocher de l'optimum d'une fonction en combinant les gènes contenus dans les différents individus de la population [Dorigo, 1992].

b) Principe

Les algorithmes génétiques classiques introduits par Holland s'appuient fortement sur un codage universel sous forme de chaînes '0' et '1' de longueur fixe et un ensemble d'opérateurs génétiques : les sélections, les crossing over ou recombinaison et les mutations. Un individu sous ce codage, appelé un chromosome, représente une configuration du problème. Les opérateurs « génétiques » sont définis de manière à opérer aléatoirement sur un ou deux individus sans aucune connaissance sur le problème.

La génétique a mis en évidence l'existence de plusieurs opérateurs au sein d'un organisme donnant lieu au brassage génétique. Ces opérations interviennent lors de la phase de reproduction lorsque les chromosomes de deux organismes fusionnent. Ces opérations sont imitées par les algorithmes génétiques afin de faire évoluer les populations de solutions de manières progressives.

a) Les sélections :

Pour déterminer quels individus sont plus enclins à obtenir les meilleurs résultats, une sélection est opérée. Ce processus est analogue à un processus de sélection naturelle, les individus les plus adaptés gagnent la compétition de la reproduction tandis que les moins adaptés meurent avant la reproduction, ce qui améliore globalement l'adaptation.

b) Les crossing over ou recombinaison :

Lors de cette opération, deux chromosomes s'échangent des parties de leurs chaînes, pour donner de nouveaux chromosomes. Ces crossing over peuvent être simples ou multiples. Dans le premier cas, les deux chromosomes se croisent et s'échangent des portions d'ADN en un seul point. Dans le deuxième cas, il y a plusieurs points de croisement. Pour les algorithmes génétiques, c'est cette opération qui est prépondérante. Sa probabilité d'apparition lors d'un croisement entre deux chromosomes est un paramètre de l'algorithme génétique. En règle générale, on fixe la proportion d'apparition à 0.7.

c) Les mutations :

D'une façon aléatoire, un gène peut, au sein d'un chromosome être substitué à un autre. De la même manière que pour les crossing over, on définit ici un taux de mutation lors des changements de populations qui est généralement compris entre 0.001 et 0.01. Il est nécessaire de choisir pour ce taux une valeur relativement faible de manière à ne pas tomber dans une recherche aléatoire et conserver le principe de sélection et d'évolution. La mutation sert à éviter une convergence prématurée de l'algorithme.

d) Codage :

Pour les algorithmes génétiques, un des facteurs les plus importants, est la façon dont sont codés les solutions, c'est-à-dire les structures de données qui coderont les gènes.

i. *Codage binaire :*

Le principe est de coder la solution selon une chaîne de bit. Ce type de codage est le plus utilisé car il présente plusieurs avantages [Holland, 1975]. Il existe au moins un coté négatif qui fait que d'autres existent. Ce codage est peu naturel par rapport à un problème donné.

ii. *Codage à caractère multiple :*

Ce type de codage est plus naturel que le codage binaire. Il est utilisé dans de nombreux cas poussés [Chen & Smith, 1996].

iii. *Codage sous forme d'arbre :*

Ce codage utilise une structure arborescente avec une racine de laquelle peuvent être issus un ou plusieurs fils. Un de leurs avantages est qu'ils peuvent être utilisés dans le cas de problèmes ou les solutions n'ont pas une taille finie. Les arbres de tailles quelconques peuvent être formés par le biais de crossing over et de mutations. Le problème de ce type de codage est que les arbres résultants sont souvent difficiles à analyser et que l'on peut se retrouver avec des arbres dont la taille est importante. Pour le choix du type de codage, il suffit de choisir celui qui semble le plus naturel en fonction du problème à traiter et développer ensuite l'algorithme de traitement.

Bien que les algorithmes génétiques soient considérés aujourd'hui comme une méthode d'optimisation, l'objectif initial consistait à concevoir des systèmes

I.5.2.2.1.3 Les méta- heuristiques éthologiques : Colonies de fourmis (Ant Colony)

Historique :

Au début de l'intelligence artificiel la démarche courante essayait de reproduire le raisonnement humain. La cognition été une activité individuelle sans l'influence du groupe au moment de son exercice. Devant la complexité des problèmes informatique cette approche n'a pas résistée, d'où la naissance de l'intelligence artificielle distribuée mettant l'accent sur une résolution distribuée des problèmes. Elle enrichit le processus de résolution de problèmes en le partageant entre plusieurs agents. Un agent est une entité autonome au comportement statique, qui peut interagir avec son environnement et éventuellement échanger de l'information avec d'autres agents.

L'essaim est un groupement d'agents dans lequel l'échange d'information influe sur le comportement individuel permettant la réalisation d'objectifs globaux hors de portée d'un agent. L'intelligence en essaim est un domaine qui tente d'utiliser la connaissance que les éthologistes ont sur les capacités collectives de résolutions de problèmes des insectes sociaux pour l'appliquer aux techniques artificielles. Elle ajoute au comportement individuel, jugé insuffisant, l'influence du groupe. Le système sera composé d'un ensemble d'entités ou agents avec une intelligence très limitée qui sont en forte interaction, l'espérance est que le comportement global émerge à partir de ces interactions. Donc l'intelligence est construite par émergence. Le comportement du groupe ne peut être déduit des règles de comportement d'un individu, réciproquement, l'individu n'a aucune notion du comportement du groupe.

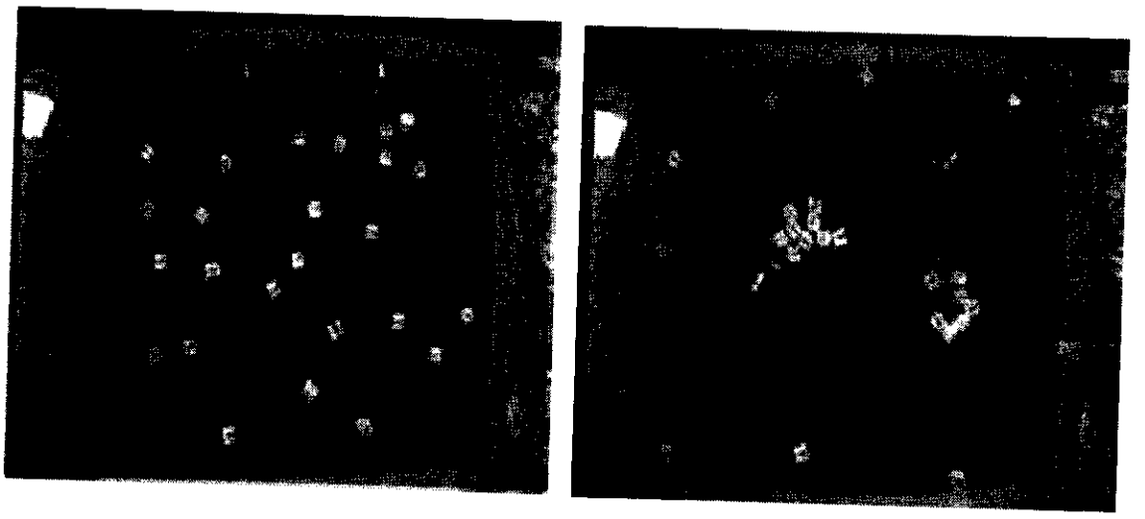
Nous savons aujourd'hui que les insectes sont dotés d'une intelligence très simple mais capable de résoudre des problèmes complexes en coopérant leurs comportements, qui semble même, du point de vue d'un observateur extérieur, résulter d'une intelligence avancée. On dit que le comportement global des insectes est construit par émergence à partir des interactions des comportements simples de chaque insecte. D'après Morin : «On peut appeler émergences les qualités ou

propriétés d'un système qui présentent un caractère de nouveauté par rapport aux qualités ou propriétés des composantes considérées isolément ou agencées différemment dans un autre type de système »

Parmi les exemples d'émergence de comportement on peut citer les travaux de Maris qui étudient le comportement d'un groupe de robots dans une enceinte fermée. Chaque robot possédait deux détecteurs d'obstacles à portée très limitée. Les robots suivaient des règles très simples :

- Si un obstacle ou un mur est à gauche, tourner à droite.
- Si un obstacle ou un mur est à droite, tourner à gauche.
- Si un obstacle est à droite et un autre à gauche, ou qu'il n y a pas d'obstacle à droite ou à gauche, continuer tout droit.
- Si un mur est heurté : tourné à droite ou à gauche au hasard.

La figure illustre la configuration initiale qui est composé de 3 robots et 25 obstacles



Départ *Résultat*

Figure I.4 : exemples d'émergence de comportement d'un groupe de robots

Le résultat est que le comportement collectif des robots été de "nettoyer" l'enceinte des obstacles. Ils les groupaient "sans le savoir". Ce résultat final n'était pas prévisible à partir des comportements individuels. Il s'agit d'un comportement émergeant de l'ensemble des actions individuelles.

Colonies de fourmis

Cette métaheuristique s'inspire des comportements collectifs des fourmis dans leurs découvertes de nouvelles sources de nourriture [Martello & Osman & Roucairol, 1999] : en effet ces insectes utilisent des phéromones afin de marquer les informations qu'ils ont recueillies sur leur environnement. On appel cela *stigmergie*.
Les algorithmes de colonies de fourmis forment une classe des méta-heuristiques récemment proposée pour résoudre des problèmes d'optimisation difficiles. Cette colonie d'agents simples (les fourmis) communiquent indirectement via des modifications dynamiques de leur environnement (les pistes de phéromones) et construisent ainsi une solution à un problème en s'appuyant sur leur expérience collective.

Le premier algorithme de ce type (Ant System) a été conçu pour le problème du voyageur de commerce. On rencontre également une application pour le problème de coloration des graphes. Des résultats de bonne qualité ont été obtenus.
L'utilisation de ces phéromones leurs permettent de repérer le plus courts chemin entre une source de nourriture et leur nid. Car malgré leur capacité cognitive limitée, elles sont collectivement capables de résoudre des problèmes complexes.

I.6 Conclusion

Les méthodes de résolution sont extrêmement nombreuses, elles sont basées sur des principes totalement différents, chacune explore et exploite l'espace de recherche selon des techniques qui lui sont propres. Comparer ces méthodes entre elles n'est pas une chose facile. Toutefois, il n'existe pas de méthodes de recherche qui soit véritablement plus performante qu'une autre sur l'ensemble des problèmes.
Pendant plusieurs décennies, les problèmes combinatoires ont constitué un sujet privilégié. Motivées par leurs applications industrielles concrètes, plusieurs équipes de recherche ont étudié ces problèmes et ont proposés divers modèles et approches de résolution.

Les métaheuristiques constituent une classe de méthodes approchées adaptables à un très grand nombre de problèmes combinatoires. Elles ont révélé leur grande efficacité pour fournir des solutions approchées de bonne qualité pour un grand nombre de problèmes d'optimisation de grande taille.

Les colonies de fourmis présentent des caractéristiques de flexibilité particulièrement intéressantes sur ces problèmes.

Pour notre étude, nous avons retenu l'approche des colonies de fourmis parce qu'elle est très performante dans de nombreux domaines. C'est une méthode très efficace lorsqu'il s'agit d'exploiter une zone de l'espace de recherche. D'autre part, elle s'adapte assez bien au problème posé.

II.2 Les fourmis réelles

Les *fourmis* famille des *formicidés* sont des insectes sociaux formant des colonies, appelées fourmilières dont l'organisation et la gestion sont parfois extrêmement complexes, contenant de quelques dizaines à plusieurs millions d'individus. Certaines espèces forment des « colonies de colonies ». Les fourmis sont classées dans l'ordre des *hyménoptères* les spécialistes estiment la présence des fourmis sur notre planète à 100×10^6 années [Monmarché, 2000].

Les fourmis constituent à l'heure actuelle un support majeur pour les théories développées en écologie comportementale et en sociobiologie. On peut citer plusieurs raisons à cet engouement :

- L'influence des fourmis sur leur environnement naturel est extrêmement importante. Il a par exemple été montré qu'elles déplacent plus de terre en foret tropicale que les vers de terre, ou encore que le poids total des fourmis sur terre est du même ordre de grandeur que le poids des humains. De plus, la domination des fourmis est une preuve de leur adaptation à des environnements très variés: Dans la forêt vierge amazonienne du Brésil, le poids sec de l'ensemble des fourmis est environ quatre fois supérieur à celui de tous les vertébrés (mammifères, oiseaux, reptiles et amphibiens) réunis [Holldobler & Wilson, 1996]. On trouve ainsi des fourmis dans tous les écosystèmes terrestres situés entre les deux cercles polaires. La connaissance de leur mode de vie est donc primordiale à la compréhension des espèces animales et végétales qui les côtoient.

- L'étude des fourmis se fait assez facilement en laboratoire car elles s'adaptent sans trop de difficultés à des environnements différents de leurs habitats d'origine.

- Les fourmis possèdent une gamme de comportements très variés, collectifs ou individuels.

II.2.1 Les insectes sociaux

La place des fourmis dans l'étude des sociétés animales est centrale car elles ont des formes très avancées de socialité.Le nombre d'espèces sociales (environ 13 500 connues [Holldobler & Wilson, 1996]) est assez réduit par rapport au nombre d'espèces d'insectes répertoriées, soit environ 750 000, alors que les insectes sociaux représentent la moitié de la biomasse des insectes. La grande diversité des fourmis (environ 10 000 espèces connues [Holldobler & Wilson, 1996]) propose une large variété de morphologies et de comportements. L'étude des fourmis, la *myrmécologie*, est donc un vaste et passionnant champ d'investigation.

Une fourmilière peut aussi être assimilée à un super organisme s'apparentant à un organisme vivant composé de cellules. Chaque cellule remplit un rôle précis tout comme les fourmis accomplissent certaines taches pour la survie et le développement du nid.

Les parallèles entre ces deux types de systèmes sont étonnants et l'étude de leur développement, la *morphogenèse* pour un organisme et la *sociogenèse* pour une société d'insectes, permet de faire certains rapprochements. L'étude de la *sociogenèse* a l'avantage d'être plus facile à réaliser puisque l'on peut retirer et injecter des constituants sans mettre en péril le développement du super organisme.

II.2.2 L'intelligence collective des fourmis

De par la grande diversité des écosystèmes colonisés (des forets vierges aux déserts), les fourmis offrent une grande diversité de comportements et de morphologies [Monmarché, 2000]. L'étude précise de leur comportement (*l'éthologie*) est souvent limitée aux espèces les moins populeuses pour des raisons pratiques évidentes d'étude en laboratoire. Cette diversité exubérante est une mine d'inspiration fascinante pour les systèmes informatiques. C'est ainsi que les capacités des fourmis en matière de coopération, de communication, de compétition et d'apprentissage, entre autres, peuvent être mises à profit pour la conception de robots ou d'algorithmes de résolution de problèmes.

II.2.2.1 La communication

Les insectes sociaux en général, et les fourmis en particulier, ont développé des mécanismes de communication très élaborés [Vander Meer & Breed , 1998]. Il a été défini douze types de réponse mettant en œuvre une forme de communication [Brossut, 1996] :

1. L'alarme ;
2. L'attraction simple ;
3. Le recrutement (pour une source de nourriture ou un site de nidification);
4. L'entretien et la mue ;
5. La trophallaxie (échange de liquides) ;
6. L'échange d'aliments solides ;
7. Les effets de groupe (augmentation ou inhibition d'une activité) ;
8. La reconnaissance des apparentés ou de caste ;
9. La détermination de caste ;
10. La compétition pour la reproduction ;
11. Le marquage du territoire et du nid ;
12. La reproduction (différenciation du sexe, de l'espèce, de la colonie...).

La communication chimique est de loin la plus présente chez les fourmis. Les phéromones (mélange d'hydrocarbures) sont à la base de la communication de nombreuses espèces. La *chémoréception* présente les avantages suivants :

- La diversité des molécules pouvant intervenir permet de fournir des informations qualitatives ;
- La stabilité du signal pour une molécule peu volatile permet d'assurer une certaine permanence.

Par contre, les principaux inconvénients de la communication chimique sont les suivants :

- Elle n'offre que peu d'informations sur la direction ;
- Sa propagation est relativement lente et elle est peu adaptée pour la transmission de messages urgents.

Les ouvrières sont par exemple capables de déposer des traces chimiques sur le trajet qu'elles empruntent pour ramener de la nourriture. Au delà du fait que ce marquage leur permet de retrouver leur chemin jusqu'à la fourmilière pour ce qui est du retour et jusqu'à la source de nourriture pour ce qui est d'exploiter une source abondante, cela leur permet de transmettre à leur congénères l'emplacement de l'aubaine.

La communication chimique est aussi mise à l'œuvre pour déclencher des alarmes quand le nid est attaqué et ainsi mobiliser un grand nombre d'individus pour défendre la fourmilière.

Ces deux mécanismes font partie des comportements de recrutement. De plus, plusieurs phéromones peuvent être utilisées et avec des concentrations différentes, constituant ainsi une sorte de langage chimique. Les principales manifestations du recrutement sont la recherche de nourriture, la construction du nid, la défense de la colonie et la migration vers de nouveaux sites de nidification.

Bien que peu répandue, certaines espèces ont une forme de communication acoustique soit en utilisant un grattoir ou en utilisant les vibrations du sol. Les mouvements peuvent aussi servir de canal de communication : certaines fourmis tisserandes se livrent à une sorte de danse pour recruter des ouvrières. On trouve aussi des ouvrières qui transportent d'autres ouvrières pour leur indiquer le nouvel emplacement du nid [Holldobler & Wilson, 1996]. La communication tactile entre aussi en jeu dans de nombreux rituels d'invitation et de recrutement. Enfin, la communication visuelle est assez difficile à mettre en évidence mais certaines espèces semblent utiliser ce canal pour déclencher des mouvements collectifs notamment lors de l'attaque de proies.

L'utilisation des phéromones est majoritairement une forme indirecte puisque l'échange d'information se fait grâce au support du sol. Quand deux individus interagissent indirectement en modifiant l'environnement on parle de stigmergie. Ce terme a été introduit par Grassé à propos des mécanismes collectifs de construction du nid chez les termites.

Les différentes applications informatiques qui découlent des capacités de communication des fourmis se retrouvent par exemple en optimisation combinatoire ou la coopération *stigmergétique* s'applique parfaitement à la recherche du plus court chemin dans un graphe.

II.2.2.2 La division du travail

Une des caractéristiques particulièrement intéressante est la capacité des sociétés d'insectes à se partager le travail. Les taches que doivent accomplir les ouvrières sont en effet multiples :

- La recherche de nourriture ;
- La défense du nid ;
- L'entretien et la construction du nid ;
- L'entretien des larves et leur approvisionnement en nourriture.

Toutes ces activités, dont l'importance est variable dans le temps et l'espace, doivent être assurés simultanément pour la survie et le développement de la colonie. C'est essentiellement la plasticité de l'organisation déployée par les fourmis qui nous intéresse. Il a été mis en évidence que certains groupes d'individus se spécialisent dynamiquement pour une tache particulière. Cette dynamique peut être mise en œuvre pour un individu particulier : sa tache de prédilection varie dans le temps, dans ce cas toutes les ouvrières sont potentiellement capables d'accomplir n'importe quelle tâche. On trouve aussi des spécialisations morphologiques, avec par exemple des variations de taille de un à dix à l'intérieur de la même espèce. Dans ce cas la dynamique est assurée par un contrôle des naissances sur chaque type de morphologie.

II.2.2.3 La construction du nid

L'architecture des nids construits par les fourmis est un exemple frappant de structure complexe. L'intérêt pour des modèles pouvant expliquer l'apparition de telles structures provient encore une fois de l'organisation distribuée qui est sous-

jacente. Il n'y a pas, a priori, de contrôle centralisé, de coordination de niveau supérieur à l'individu. La structure émerge des interactions inter- individuelles et avec l'environnement.

Les fourmis tisserandes sont par exemple capables d'unir leurs efforts pour rapprocher des feuilles en formant de véritables ponts puis d'unir les bords des feuilles en utilisant la soie produite par leurs larves. La construction du nid chez les termites a été étudiée par Deneubourg. L'apparition des piliers dans une termitière pouvant être expliquée par l'amplification de multiples fluctuations chaotiques.

II.2.2.4 La quête de nourriture

La recherche de la nourriture (le fourragement) est une activité souvent plus dispersée spatialement que la construction du nid et qui peut aussi être mise en œuvre de façon très différente suivant les espèces de fourmis. Les stratégies de recherche de nourriture sont en effet extrêmement diversifiées. Par exemple à cause des différences de régime alimentaire : certaines espèces peuvent être spécialisées sur un unique type d'aliment. On peut aussi trouver des mécanismes très élaborés comme la culture de champignon ou l'élevage de pucerons. La recherche de nourriture est une activité risquée (les fourrageuses de Cataglyphis bicolor ont par exemple une espérance de vie de 6.1 jours [Holldobler & Wilson, 1996]) mais souvent efficace (la quantité de nourriture ramenée au nid au cours d'une vie représente de 15 à 20 fois le poids d'un individu). La communication peut avoir un impact important, en particulier pour les mécanismes de recrutement dont le principal intérêt collectif est de rassembler les ouvrières sur les sources de nourriture rentables. D'un point de vue plus général, la communication mise en œuvre pour la recherche de nourriture peut être considérée comme une forme de mémoire collective quand elle s'appuie sur la modification de l'environnement telle que l'utilisation des phéromones.

II.2.2.5 Capacités individuelles

Les capacités individuelles des fourmis peuvent servir de modèle à des systèmes. Nous citons par exemple les points suivants :

* Individuellement, une fourmi possède certaines capacités d'apprentissage, et notamment quand elle se déplace autour du nid. Les expériences de Schatz et ses collègues montrent que les fourmis du genre *Cataglyphis* sont capables d'apprendre visuellement des routes familières pour se déplacer entre un site alimentaire et leur nid;

* Du point de vue physique, certaines espèces ont des capacités étonnantes comme les fourmis Gigantiops destructor capables de faire des bonds impressionnants et dotées de capacités visuelles inhabituelles ce qui les a rendues difficiles à observer.

La plupart des caractéristiques qui intéressent l'informatique sont cependant collectives. Les caractéristiques individuelles ne sont évidemment pas une particularité des fourmis mais de tous les organismes vivants ayant un souci de survie.

II.2.3 Les comportements collectifs des fourmis

II.2.3.1 L'auto organisation chez les insectes sociaux

Les théories rassemblées sous le terme d'auto organisation ont originellement été développées en physique ou en chimie pour décrire l'émergence de phénomènes macroscopiques à partir d'interactions et de processus définis au niveau microscopique. L'auto organisation se prête bien à l'étude des insectes sociaux qui montrent des comportements collectifs complexes issus de comportements individuels simples. On peut regrouper les processus d'auto organisation chez les insectes sociaux en quatre groupes tant leur diversité est importante :

- Le recrutement et l'exploitation collective de sources de nourriture : le fourragement met à jour des stratégies qui permettent aux insectes une grande adaptation à leur milieu ;
- La division du travail et l'organisation des rôles sociaux : à l'intérieur d'une même société, on peut observer différentes castes spécialisées dans un certain nombre de taches (élevage du couvain, recherche de nourriture, construction du nid, ...) ;
- L'organisation de l'environnement : la construction du nid est un symbole de l'organisation distribuée des insectes. Le nid est construit sans que les insectes soient dirigés, ils répondent à un certain nombre de stimuli provenant de leur environnement ;
- La reconnaissance inter-individuelle : chaque fourmi est capable d'identifier ses congénères tout en participant elle même à l'identité de sa colonie (l'échange d'aliments entre les individus d'une même colonie, la trophallaxie, est un exemple d'acte altruiste permettant d'homogénéiser l'identité de la colonie). Les explications du mécanisme de reconnaissance ne sont pas encore parfaitement établies. Cependant, il s'avère qu'il y ait à la fois une composante génétique et une composante acquise par apprentissage. Un réseau de neurones a par exemple été utilisé pour reproduire ce mécanisme d'apprentissage puis de différenciation entre les composés chimiques, dans le cas des termites.

Ces processus d'auto-organisation sont à l'origine de ce que l'on dénomme l'intelligence collective. On parle d'intelligence collective quand un groupe social peut résoudre un problème dans un cas ou un agent isolé en serait incapable.

II.2.3.2 Stigmergie

La stigmergie est un des concepts à la base de la création des méta- heuristiques de colonies de fourmis. Elle est précisément définie comme une «forme de communication passant par le biais de modifications de l'environnement», mais on peut rencontrer le terme « interactions sociales indirectes » pour décrire le même phénomène. La spécificité de la stigmergie est que les individus échangent des

informations par le biais du travail en cours et de l'état d'avancement de la tache globale à accomplir [di Caro & Dorigo, 1998].

II.2.3.3 Contrôle décentralisé

Dans un système auto-organisé, il n'y a pas de prise de décision à un niveau donné suivie d'ordres et d'actions prédéterminées. En effet, dans un système décentralisé; chaque individu dispose d'une vision locale de son environnement, et ne connaît donc pas le problème dans son ensemble. La littérature des systèmes multi-agents emploie souvent ce terme (Contrôle décentralisé) ou celui « d'intelligence artificielle distribuée », bien que, d'une manière générale, l'étude des systèmes multi agents tende à utiliser des modèles de comportement plus complexes, fondé notamment sur les sciences de la cognition. Les avantages d'un contrôle décentralise sont notamment la robustesse et la flexibilité : systèmes robustes, car capables de continuer à fonctionner en cas de panne d'une de leurs composantes. Flexibles ; car efficaces sur des problèmes dynamiques.

II.2.3.4 Hétérarchie dense

L'hétérarchie dense est un concept issu directement de la biologie, utilisé pour décrire l'organisation des insectes sociaux, et plus particulièrement des colonies de fourmis [Holldobler & Wilson, 1996]. La définition peut être traduite comme suit :
Une colonie de fourmis est une variante particulière de hiérarchie qui peut avantageusement être appelée une hétérarchie. Cela signifie que les propriétés des niveaux globaux agissent sur les niveaux locaux, mais que l'activité induite dans les unités locales influence en retour les niveaux globaux. L'hétérarchie est dite dense dans le sens ou un tel système forme un réseau hautement connecte, ou chaque individu peut échanger des informations avec n'importe quel autre. Ce concept est en quelque sorte opposé à celui de hiérarchie. La reine gouvernerait ses sujets en faisant passer des ordres dans une structure verticale, alors que, dans une hétérarchie, la structure est plutôt horizontale Figure II.1.

Figure II.1 : [a] Hiérarchie, [b] Hétérarchie dense : deux concepts opposés.

On constate que ce concept recoupe celui de contrôle décentralisé, mais aussi celui de stigmergie, en ce sens que l'hétérarchie décrit la manière dont le flux d'information parcourt le système. Cependant, dans une hétérarchie dense, tout type de communication doit être pris en compte, tant la stigmergie que les échanges directs entre individus.

II.2.3.5 Les pistes de phéromones

Les fourmis ont la particularité d'employer pour communiquer des substances volatiles appelées phéromones. Elles sont attirées par 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 former ainsi des pistes odorantes, qui pourront être suivies par leurs congénères Figure II.2.
Les fourmis utilisent les pistes de phéromone 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 à exploiter, sans que les individus aient une vision globale du trajet.

En effet, comme l'illustre la Figure II.2, les fourmis le plus rapidement arrivées au nid, après avoir visite la source de nourriture, sont celles qui empruntent les/le chemin le plus court. Ainsi, la quantité de phéromone présente sur le plus court trajet est légèrement plus importante que celle présente sur le chemin le plus long. Or, une piste présentant une plus grande concentration en phéromone est plus attirante pour les fourmis, elle a une probabilité plus grande d'être empruntée. La piste courte va alors être plus renforcée que la longue, et, à terme, sera choisie par la grande majorité des fourmis.

On constate qu'ici le choix s'opère par un mécanisme d'amplification d'une fluctuation initiale. Cependant, il est possible qu'en cas d'une plus grande quantité de phéromone déposée sur les grandes branches, au début de l'expérience, la colonie choisisse le plus long parcours.

D'autres expériences, avec une autre espèce de fourmis, ont montré que si les fourmis sont capables d'effectuer des demi-tours sur la base d'un trop grand écart par rapport à la direction de la source de nourriture, alors la colonie est plus flexible et le risque d'être piégé sur le chemin long est plus faible.

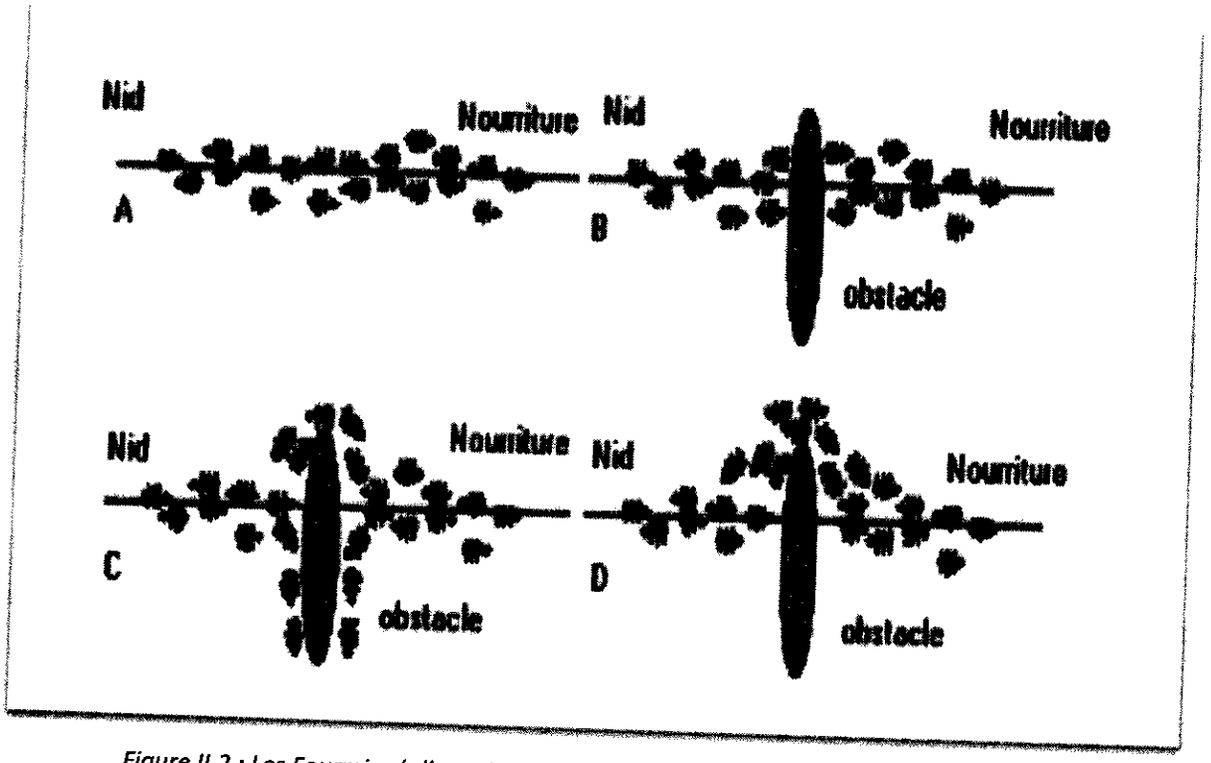


Figure II.2 : Les Fourmis réelles suivent un chemin entre le Nid et la Nourriture.

Les Fourmis réelles suivent un chemin entre le Nid et la Nourriture. (B) Un obstacle apparaît sur le chemin : Les Fourmis choisissent de tourner soit à gauche soit à droite avec une probabilité égale. (C) La phéromone est déposé plus rapidement sur le chemin le plus court. (D) Toutes les fourmis ont choisi le chemin le plus court.

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

II.3 Les fourmis artificielles

La fourmi artificielle se présente sous la forme d'un ensemble de procédures qui définissent son comportement [Di Caro & Dorigo, 1998]. Celui-ci est très semblable à celui de la fourmi naturelle quand elle recherche de la nourriture Figure II.2. Dans ce cas, une fourmi n'a qu'un rôle assez simple qui consiste à se déplacer du nid jusqu'à la source de nourriture et à y revenir. Le code qui définit leur comportement permet aux fourmis artificielles de se déplacer dans l'espace combinatoire formé par les différents éléments qui peuvent être utilisés pour le problème à résoudre. Pour utiliser un vocabulaire informatique, nous dirons qu'elles construisent une solution. La mémorisation de ces déplacements donne la forme d'une solution où chaque étape est désignée par l'indice de l'élément et où l'ordre du parcours désigne la position des éléments dans la solution.

En se basant sur les définitions précédentes, il est possible de décrire le comportement des fourmis virtuelles en tant qu'agents :

1. Qui est capable d'agir dans un environnement;
2. Qui peut communiquer directement avec d'autres agents;
3. Qui, est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser) ;
4. Qui possède des ressources propres;
5. Qui, est capable de percevoir (mais de manière limitée) son environnement;
6. Ne dispose que d'une représentation partielle de cet environnement;
7. Qui, possède des compétences et offre des services;
8. Qui, peut éventuellement se reproduire;
9. Dont le comportement tend à, satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

En se basant sur ces définitions, il est possible de décrire le comportement des fourmis virtuelles en tant qu'agents :

Les entités informatiques que nous venons de définir sont dites virtuelles car elles n'ont pas d'existence matérielle au contraire des entités physiques qui interagissent dans le monde concret (Des exemples de ces agents physiques sont un robot, un avion, une voiture);

Les agents sont capables d'agir : dans le cas des fourmis virtuelles, elles modifient les valeurs de phéromone associées aux différents éléments. Par cette action elles changent leur environnement, ce qui influera sur le choix des autres fourmis à l'itération suivante;

Les agents sont capables de communiquer : les fourmis utilisent comme on l'a vu précédemment la phéromone comme médium de communication indirecte;

Les agents sont doués d'autonomie : chaque fourmi â pour but de construire une solution pour un problème donné, se contentant pour cela d'appliquer les règles de sélection qui définissent son comportement, la fourmi utilise la phéromone et parfois des valeurs heuristiques;

Les agents n'ont qu'une représentation partielle de leur environnement : lors de la construction d'une solution, la fourmi ne connaît à chaque étape que les éléments qu'elle a déjà choisis et les valeurs de phéromone correspondant aux éléments qui pourront l'être.

Comme nous avons pu le voir, les fourmis peuvent résoudre collectivement des problèmes complexes. Un des exemples marquant est celui de la découverte du chemin le plus court dans l'expérience du choix entre les deux chemins. Chacune des fourmis, isolement, ne peut trouver la meilleure solution. C'est grâce au mécanisme d'auto-organisation qui émerge de la communication indirecte (stigmergie) que le plus court chemin peut être découvert. Le modèle de la stigmergie peut être facilement étendu aux agents artificiels en associant aux éléments d'un problème des variables d'état spécifique, qui serviront de support à la communication indirecte entre fourmis.

Pour simuler ce mécanisme, il faut intégrer un processus de renforcement positif. Dans le cas des fourmis, plus le chemin est court, plus les fourmis le parcourront vite et plus la quantité de phéromone associée sera élevée. Afin de pouvoir comparer les solutions, il est nécessaire d'introduire une mesure qui donne l'adéquation de la solution au problème : c'est la fonction qualité (en anglais : fitness). La valeur de cette qualité est utilisée dans le calcul de la mise à jour des valeurs associées aux éléments du problème lors de la phase de renforcement. Pour la recherche du chemin le plus court, on peut prendre tout simplement la distance comme mesure de qualité : plus le chemin est court, plus il sera renforcé.

Une des propriétés de la phéromone est son caractère volatil. Dans le modèle virtuel, un mécanisme similaire sera utilisé afin d'éviter une convergence prématurée (stagnation) due à la découverte d'un optimum local.

Entre ces deux modèles, nous passons d'un univers continu à un univers discret. Ainsi, les fourmis virtuelles sautent d'un élément à un autre, tandis que les fourmis naturelles progressent de façon continue sur le chemin. Cette discrétisation impose que l'évaporation ou la modification des valeurs de phéromone ne puisse se faire en continu. Cet ajustement de valeurs n'est souvent réalisable qu'après la construction de la solution complète.

II.4 Optimisation par colonies de fourmis

Application des colonies de fourmis au problème du voyageur du commerce

Le problème du voyageur de commerce (Travelling Salesman Problem, **TSP**) a fait l'objet de la première implémentation d'un algorithme de colonies de fourmis : le [Ant System (AS)] [Bentley, 1992].

Le problème du voyageur de commerce consiste à trouver le trajet le plus court reliant n villes données, chaque ville ne devant être visitée qu'une seule fois. Le

Ou α et β sont deux paramètres contrôlant l'importance relative de l'intensité de la piste $\tau_{ij}(t)$ et de la visibilité η_{ij} . Avec α=0, seule la visibilité de la ville est prise en compte; la ville la plus proche est donc choisie à chaque pas. Au contraire, avec β=0, seules les pistes de phéromone sont pris en compte. Pour éviter une sélection trop rapide d'un trajet, un compromis entre ces deux paramètres, jouant sur les comportements de diversification et d'intensification est nécessaire.

Après un tour complet, chaque fourmi laisse une certaine quantité de phéromone

$\Delta\tau_{ij}^{k}(t)$ sur l'ensemble de son parcours, quantité qui dépend de la qualité de la solution trouvée :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^{k}(t)} & si\ (i,j) \in T^{k}(t) \\ 0 & si\ (i,j) \notin T^{k}(t) \end{cases} \qquad (2.2)$$

Où

- $T^{k}(t)$: est le trajet effectué par la fourmi K à l'itération t.
- $L^{k}(t)$: la longueur de la tournée et Q un paramètre fixé.

L'algorithme ne serait pas complet sans le processus d'évaporation des pistes de phéromone. En effet, pour éviter d'être piégé dans des solutions sous optimales, il est nécessaire de permettre au système «d'oublier» les mauvaises solutions. On contrebalance donc l'additivité des pistes par une décroissance constante des valeurs des arêtes à chaque itération. La règle de mise à jour des pistes est donc :

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t \qquad (2.3)$$

Ou

- $\Delta\tau_{ij}(t) = \sum_{k-1}^{m} \Delta\tau_{ij}^{k}(t)$
- m est le nombre de fourmis.

La quantité initiale de phéromone sur les arêtes est une distribution uniforme d'une petite quantité. $\tau_0(0) = \varepsilon$

Pour $t = 1, \ldots\ldots, t_{max}$

  Pour chaque fourmi $k = 1, \ldots, m$

    Choisir une ville au hasard

    Pour chaque ville non visitée i

    Choisir une ville j, dans la liste j_i^k des villes restantes, selon la formule 2.1

    Fin Pour

    Déposer une piste $\Delta\, \tau_{ij}^k(t)$ sur le trajet $T^k(t)$ conformément à l'équation 2.2

  Fin pour

  Evaporer les pistes selon la formule 2.3

Fin pour

Algorithme (2.1) : Algorithme de l'Ant System sur un problème de voyageur de commerce.

La figure II.3 présente un exemple simplifié de problème du voyageur de commerce.

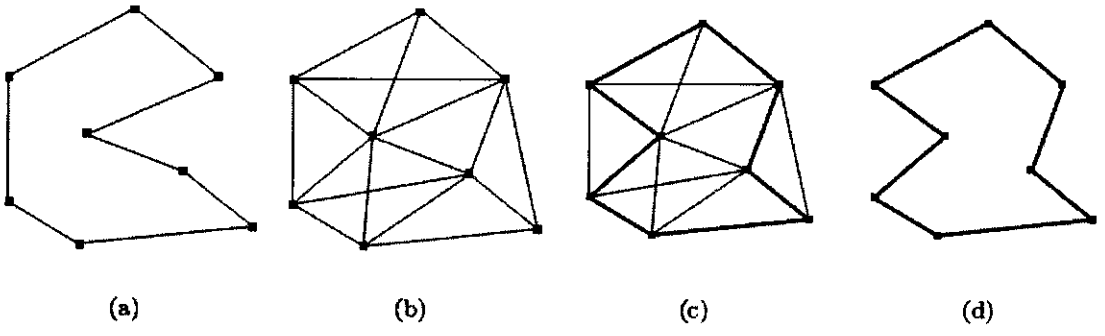


Figure II.3 : Le problème du voyageur du commerce, les points représente les villes et l'épaisseur des arêtes la quantité de phéromone déposée. (a) exemple de trajet construit par une fourmi. (b) au début du calcul, tous les chemins sont explorés. (c) le chemin le plus court est plus renforcé que les autres. (d) l'évaporation permet d'éliminer les moins bonne solutions.

II.4.2 Variantes

II.4.2.1 Ant System & élitisme

Une première variante du « Système de Fourmis » a été proposée : elle est caractérisée par l'introduction de fourmis « élitistes ». Dans cette version, la meilleure fourmi (celle qui a effectué le trajet le plus court) déposé une quantité de phéromone plus grande, dans l'optique d'accroître la probabilité des autres fourmis d'explorer la solution la plus prometteuse.

II.4.2.2 Ant-Q

Dans cette variante de AS, la règle de mise à jour locale est inspirée du « Q learning ». Cependant, aucune amélioration par rapport a l'algorithme AS n'a pu être démontrée. Cet algorithme n'est d'ailleurs, de l'aveu même des auteurs, qu'une préversion du « Ant Colony System ».

II.4.2.3 Ant Colony System

L'algorithme « Ant Colony System » (ACS) [Dorigo & Di Caro & Gambardella, 1999] [Dorigo & Gambardella, 1997] a été introduit pour améliorer les performances du premier algorithme sur des problèmes de grandes tailles. ACS est fondé sur des modifications du AS :

ACS introduit une règle de transition dépendant d'un paramètre $q_0 (0 \leq q_0 \leq 1)$, qui définit une balance diversification/intensification. Une fourmi k sur une ville i choisira une ville j par la règle :

$$i = \begin{cases} argmax_u \, j_i^k [(\tau_{iu}(t))(\eta_{ij})^\beta] & si \; q \leq q_0 \\ J & si \; q > q_0 \end{cases} \qquad (2.4)$$

Ou q est une variable aléatoire uniformément distribuée sur [0,1] et $J \in J_i^k$ une ville sélectionnée aléatoirement selon la probabilité :

$$P_i^k J(t) = \frac{\left(\tau_{ij}(t)\right)(\eta_{ij})^p}{\sum_{l \in j_i^k} (\tau_{il}(t))(\eta_{il})^\beta} \qquad (2.5)$$

En fonction du paramètre q_0, il y a donc deux comportements possibles : si $q > q_0$ le choix se fait de la même façon que pour l'algorithme *AS*, et le système tend à effectuer une diversification; si $q \leq q_0$, le système tend au contraire vers une intensification. En effet, pour $q \leq q_0$, l'algorithme exploite davantage l'information récoltée par le système, il ne peut pas choisir un trajet non exploré.

La gestion des pistes est séparée en deux niveaux : une mise à jour locale et une mise à jour globale. Chaque fourmi dépose une piste lors de la mise à jour locale :

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\tau_0 \qquad (2.6)$$

Où τ_0 est la valeur initiale de la piste. A chaque passage, les arêtes visitées voient leur quantité de phéromone diminuer, ce qui favorise la diversification par la prise en compte des trajets non explorés. A chaque itération, la mise à jour globale s'effectue comme ceci :

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad (2.7)$$

Où les arêtes (i,j) appartiennent au meilleur tour T^+de longueur L^+et ou $\Delta\tau_{ij}(t) = \frac{1}{L^+}$. Ici, seule la meilleure piste est donc mise à jour, ce qui participe à une intensification par sélection de la meilleure solution.

Le système utilise une liste de candidats. Cette liste stocke pour chaque ville les v plus proches voisines, classées par distances croissantes. Une fourmi ne prendra en compte une arête vers une ville en dehors de la liste que si celle-ci a déjà été explorée. Concrètement, si toutes les arêtes ont déjà été visitées dans la liste de candidats, le choix se fera en fonction de la règle (2.5) sinon c'est la plus proche des villes non visitées qui sera choisies.

II.4.2.4 Max-Min Ant System

Cette variante (notée MMAS) est fondée sur l'algorithme AS et présente quelques différences notables [Stützle and Hoos, 1997, Stützle and Hoos, 2000] :

1. Seule la meilleure fourmi met à jour une piste de phéromone.
2. Les valeurs des pistes sont bornées par τ_{min} et τ_{max}.
3. Les pistes sont initialisées à la valeur maximum τ_{max}.
4. La mise à jour des pistes se fait de façon proportionnelle, les pistes les plus fortes étant moins renforcées que les plus faibles.
5. Une réinitialisation des pistes peut être effectuée.

Les meilleurs résultats sont obtenus en mettant à jour la meilleure solution avec une fréquence de plus en plus forte au cours de l'exécution de l'algorithme.

II.4.3 Choix des paramètres

Pour l'algorithme *AS*, les auteurs préconisent que, bien que la valeur de *Q* ait peu d'influence sur le résultat final, cette valeur soit du même ordre de grandeur qu'une estimation de la longueur du meilleur trajet trouvé. D'autre part, la ville de départ de chaque fourmi est typiquement choisie par un tirage aléatoire uniforme, aucune influence significative du placement de départ n'ayant pu être démontrée.
En ce qui concerne l'algorithme ACS, les auteurs conseillent d'utiliser $\tau_0=(n\,L_{nn})^{-1}$, ou *n* est le nombre de villes et la longueur d'un tour trouvé par la méthode du plus proche voisin.

Le nombre de fourmis *m* est un paramètre important ; les auteurs suggèrent d'utiliser autant de fourmis que de villes (*m* = *n*) pour de bonnes performances. Il est possible de n'utiliser qu'une seule fourmi, mais l'effet du parallélisme naturel de l'algorithme sera alors perdu, ce qui peut s'avérer néfaste pour certains problèmes. En règle générale, les algorithmes de colonies de fourmis semblent assez peu sensibles à un réglage fin du nombre de fourmis.

II.4.4 Formalisation d'un algorithme de colonie de fourmis

La description qui a été proposée dans [Dorigo and Stützle, 2003], qui s'applique aux problèmes (combinatoires) où une construction partielle de la solution est possible. Ce cas, bien que restrictif, permet de dégager les apports originaux de ces métaheuristiques (dénommées ACO, pour "Ant Colony Optimization", par les auteurs).

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

Une formalisation plus précise existe [Dorigo and Stützle, 2003]. 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 phéromone 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.

II.4.4.1 Formalisation

II.4.4.1.1Représentation du problème

Le problème est représenté par un jeu de solutions, une fonction objective 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 associe à des états autres que des solutions [Bouri & Zeblah & Ghoraf & Hadjeri & Hamdaoui, 2005]

Dans cette représentation, les fourmis construisent des solutions en se déplaçant sur un graphe *G = (C, L)* ou 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).

II.4.4.1.2 Organisation de la méta- heuristique

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 forme d'oubli favorisant l'exploration de nouvelles régions.

II.4.4.2 Phéromones et mémoire

L'utilisation de la stigmergie est cruciale pour les algorithmes de colonies de fourmis. 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. Par exemple, pour le problème du voyageur de commerce, une implémentation efficace consiste à utiliser une piste τ_{ij} entre deux villes i et j comme une représentation de l'intérêt de visiter la ville j après la ville i. Une autre représentation possible, moins efficace en pratique, consiste à considérer τij comme une représentation de l'intérêt de visiter i en tant que jème ville. En effet, 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 stigmergie. 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 [Taillard et al., 1998], où la dynamique de stockage et de partage de l'information va être cruciale pour le système.

II.4.4.3 Intensification/diversification

Le problème de l'emploi relatif de processus de diversification et d'intensification est un problème extrêmement courant dans la conception et l'utilisation de métaheuristiques. 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 va s'agir de choisir où et quand "injecter de l'aléatoire" dans le système (diversification) et/ou améliorer une solution (intensification).

Dans les algorithmes de type ACO, comme dans la plupart des cas, 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 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. À 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. On peut également introduire des modifications de la gestion des pistes de phéromone.

Par exemple, l'emploi de stratégies élitistes (les meilleures solutions contribuent plus aux pistes, l'algorithme AS avec élitisme) favorise l'intensification, alors qu'une réinitialisation de l'ensemble des pistes favorisera l'exploration (section II.4.2.4, algorithme MMAS).

Ce choix diversification/intensification peut s'effectuer de manière statique avant le lancement de l'algorithme, en utilisant une connaissance a priori du problème, ou de manière dynamique, en laissant le système décider du meilleur réglage. Deux approches sont possibles : un réglage par les paramètres ou l'introduction de nouveaux processus.

Dans ces algorithmes fondés en grande partie sur l'utilisation de l'auto-organisation, ces deux approches peuvent être équivalentes, un changement de paramètre pouvant induire un comportement complètement différent du système, au niveau global [Dréo, 2003].

II.4.4.4 Recherche locale et heuristiques

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. Ce qui différencie une métaheuristique de type ACO intéressante d'un algorithme réellement efficace est bien souvent l'hybridation avec une recherche locale.

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.

Il faut noter que ces deux approches sont similaires de par l'emploi qu'elles font des informations de coût pour améliorer une solution. La recherche locale le fait de façon plus directe que l'heuristique, cependant que cette dernière est peut-être plus naturelle pour utiliser des informations a priori sur le problème [Dréo, 2003].

II.4.4.5 Parallélisme

La structure même des métaheuristiques de colonies de fourmis comporte un parallélisme intrinsèque. D'une manière générale, les solutions de bonnes qualités émergent du résultat des interactions indirectes ayant cours dans le système, 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 [Dréo, 2003].

II.4.4.6 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, 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 [Gutjahr, 2000, Gutjahr, 2002] : 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 $\epsilon > 0$ une "faible" valeur, l'algorithme trouvera la solution optimale avec une probabilité $P_t \geq 1-\epsilon$, après un temps $t \geq t_0$ (où t_0 est fonction de ϵ).

II.5 Conclusion

Dans ce chapitre, nous avons vu une généralité sur les métaheuristiques des colonies de fourmis et une analogie sur les fourmis naturelles, commençant par connaître leurs comportements dans la nature, ensuite l'optimisation par les colonies de fourmis et puis la formalisation de l'algorithme. Dans le chapitre suivant, nous allons présenter notre démarche pour faire la conception de notre méthode.

Chapitre III

CONCEPTION DU SYSTÈME

Chapitre III : CONCEPTION DU SYSTÈME

III.1 Introduction

Dans les chapitres précédents nous avons présenté des généralités sur les méthodes d'optimisation et un état de l'art sur les méthodes d'optimisation de colonie de fourmis, dans ce chapitre nous allons entamer la description du problème d'application que nous avons réalisé.

III.2 Conception

III.2.1 Description du problème

Étant donné un type de textes habituellement tapés (courriel, programmation, ...) on veut obtenir le clavier virtuel qui minimisera l'effort à fournir « *la fatigue* » qui est crucial dans le cas d'une personne lourdement handicapée.

III.2.2 Solution

Pour résoudre ce problème, deux étapes sont nécessaire :

- Adaptation d'un algorithme de colonie de fourmis au problème posé.
- Implémenter l'algorithme final et faire tous les tests possibles afin de trouver la solution la plus intéressante.

III.2.3 Principe général

L'algorithme que nous avons implémenté est une adaptation de l'algorithme AS :

1. chaque fourmi tente de construire un clavier à chaque itération.
2. plus une touche est phéromonée par rapport a un caractère, plus elle aura la chance à être allouée avec ce dernier.
3. plus une touche est loin de la touche précédemment allouée, moins elle aura de chance d'être choisie (c'est la « visibilité »).
4. une fois son trajet terminé, la fourmi dépose, sur l'ensemble des arêtes parcourues, plus de phéromones.

5. les pistes de phéromones s'évaporent après chaque itération.

III.2.4 Hypothèses de travail :

- l'utilisateur ne tape qu'une touche à la fois (en déplaçant un pointeur)
- l'effort visuel de recherche d'un symbole est moins important que le déplacement pour l'atteindre.
- pour étudier le problème on peut se limiter à un sous-ensemble de symboles (par exemple : pas de majuscules).
- la fatigue est uniquement liée à la distance que l'utilisateur doit faire parcourir à son pointeur.
- on se limite aux claviers rectangulaires $m \times n$.

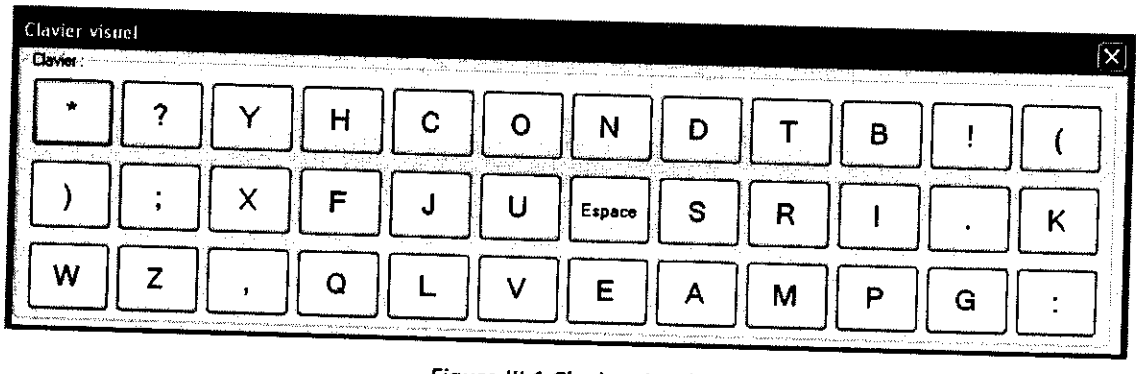


Figure III.1 Clavier visuel.

III.2.5 Les éléments nécessaires

III.2.5.1 Epuration du texte et sa distribution sur les fourmis

L'hypothèse de restriction des caractères formulé dans le cahier des charges du projet, nous a imposé l'intégration d'un module d'épuration du texte qu'on va soumettre au programme d'optimisation.

Dans ce module du programme on va :

- réduire l'ensemble des caractères au format minuscule.
- éliminer les caractères spéciaux et les chiffres.

Dans un deuxième temps, chaque fourmi prendra un segment de texte contenant un nombre précis de caractères pour construire un clavier, ce segment, dans la

majorité des cas ne contiendra pas la totalité des symboles traités, il faudra alors les allouer aléatoirement par la suite sur le clavier, pour cela, la taille du segment alloué à chaque fourmi doit être suffisamment grande pour minimiser le nombre de symboles aléatoirement alloués et au même temps pas trop grand pour permettre au traitement sur la machine de se dérouler dans des temps raisonnables, donc, la désignation d'un nombre qui permettra la satisfaction de ces deux exigences sera nécessaire.

Dans le chapitre suivant «*Testes et Résultats* » nous tenteront de quantifier ce chiffre en effectuant une série de testes.

III.2.5.2 Matrice de Phéromones

Contrairement au problème du voyageur de commerce, où les fourmis d'une itération t déposent les phéromones entre les villes pour permettre aux arrêtes de devenir plus attractive pour les fourmis des itérations suivantes, notre problématique, à cause de l'existence de deux entités (*Touche, Caractère*) à la place d'une seule (ville), nous a amené à utiliser une adaptation de l'algorithme AS.

Les fourmis qui évoluent dans notre système à chaque itération déposent une quantité de phéromones entre les ***Touches*** et les ***Caractères.***

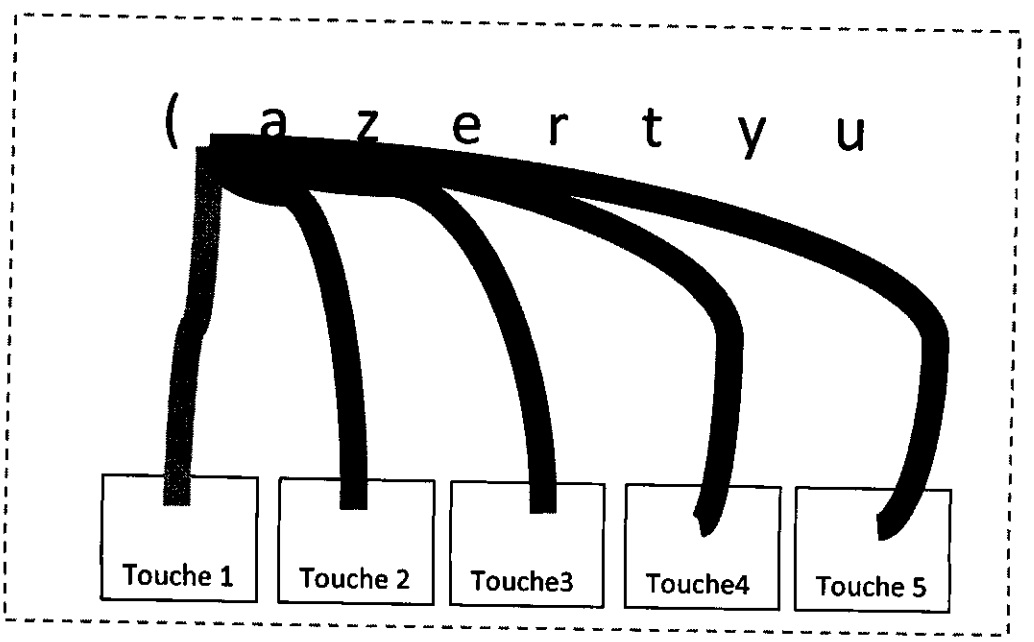


Figure III.2 Représentation des phéromones

L'ensemble des valeurs de phéromones sera sauvegardé dans une matrice « *unique* » appelée *matrice des phéromones*, qui servira par la suite pour calculer les probabilités d'allocation pour les touches restantes.

	a	b	c	d		*	?	/	!
Touche 1	0.01	0.5	5	0.5	.....	1	0.5	0.5	0.5
Touche 2	0.5	0.7	0.5	0.5	.....	5	0.3	0.2	0.5
Touche 3	0.5	0.5	0.9	0.5	.....	0.4	0.5	0.5	0.7
		.....							
		.....							
		.....							
		.....							
Touche 34	0.5	0.1	3	0.5	.....	1	0.5	0.5	0.5
Touche 35	0.5	0.5	2	6	.....	3	0.5	0.5	0.5
Touche 36	9	0.5	0.5	8	.....	3	0.5	0.5	0.5

Figure III.3 Matrice de phéromones

III.2.5.3 Calcule de la visibilité

La visibilité (*l'inverse de la distance)* entre deux villes dans un problème de voyageur de commerce, est remplacée dans notre cas par l'inverse de la distance entre la touche précédemment allouée et la touche en cours de traitement, c'est pour cela qu'après chaque allocation d'une touche, les coordonnées de cette dernière seront gardés pour permettre le calcule des probabilités pour les touches suivantes, et cela pour chaque fourmis.

Les coordonnées des touches dernièrement allouées par l'ensemble des fourmis seront gardées en mémoire dans un tableau.

	Dernière touche allouée
Fourmis 1	12
Fourmis 2	34
Fourmis 3	8
Fourmis 4	32
.	.
.	.
Fourmis 30	5

Figure III.4 représentation du vecteur des touches dernièrement allouées pour chaque fourmi dans l'itération t.

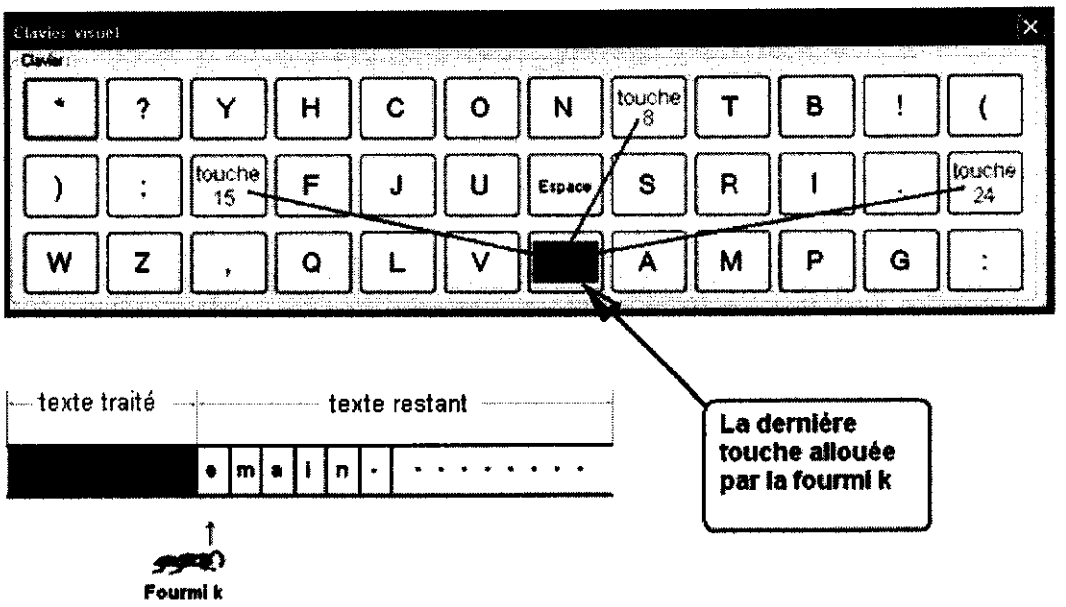


Figure III.5 représentation d'une allocation d'un symbole par une fourmi dans une itération.

Dans la *Figure III.5* la fourmi ***k*** de l'itération ***t*** va allouer le symbole « *E* » avec une touche précédemment allouée : *touche 31. (contenant le symbole « D »)* , et va procéder à un choix entre les touches vacantes de son clavier (touche 15,8 ,24).

III.2.5.4 Règle d'allocation des symboles

La règle d'allocation des symboles sur les touches est la suivante :

$$p_{it}^{k}(T) = \frac{(\tau_{it}(T))^{\alpha}.(\eta_{te_k})^{\beta}}{\sum_{l \in J}(\tau_{il}(T))^{\alpha}.(\eta_{il})^{\beta}} \qquad (3.1)$$

Où :

i : est le caractère à placer.

t : est la touche pour laquelle on veut calculer la probabilité que le caractère i lui-soit affecté.

$\tau_{it}(T)$: est la valeur de phéromone qui se trouve entre le caractère i et la touche t dans l'itération T.

e_k : est la touche dernièrement allouée par la fourmi k.

η_{te_k} : est l'inverse de la distance (la visibilité) entre la touche t et e_k.

J : est l'ensemble des touches pas encore allouées par la fourmi k dans son clavier Ck.

α, β : les paramètres contrôlant l'importance relative de l'*intensité* de la piste et de la *visibilité*.

Remarque : Il est nécessaire de régler α et β en faisant un compromis entre une intensification trop grande ($\beta = 0$), ou une diversification trop poussée ($\alpha = 0$) pour l'obtention d'un résultat satisfaisant.

III.2.5.5 L'Evaluation des claviers

L'hypothèse de travail suivant laquelle nous avons fondé notre travail stipule que l'utilisateur ne tape qu'une seule touche à la fois avec un curseur « clavier virtuel », Aussi, que l'effort de recherche des caractères sur le clavier est négligeable par rapport à l'effort fourni pour les atteindre.

Suivant cette hypothèse, L'évaluation des claviers sera calculée en fonction de la somme des distances parcourues pour saisir un texte globale Tg, suivant la règle :

$qk(T,S) = \sum_{i=0}^{|S|-1} d\varepsilon\,(S[i], S[i+1])$ (3.2)

$S = \{s[1], s[2], \ldots. ; s[|S|]\}$: est une séquence de symboles (un texte) de longueur $|S|$

$d\varepsilon(x,y)$: Distance euclidienne entre deux touche T_1 et T_2 de coordonnées (x_1, x_2) et (y_1, y_2) sur le clavier *Ck* :

$$d\varepsilon(x,y) = \sqrt{(x1-y1)^2 + (x2-y2)^2}$$

III.2.6 Elitisme :

Dans notre projet on a utilisé une variante du « As » appelé élitisme qui a été proposée dans [Dorigo & Colorni & Maniezzo, 1996] : cette variante est caractérisée par l'introduction des fourmis "élitistes". Dans cette version, la meilleure fourmi (celle qui a effectué le trajet le plus court) dépose une quantité de phéromone plus grande, dans l'optique d'accroitre la probabilité des autres fourmis d'explorer la solution la plus prometteuse.

Dans notre cas, la fourmi ayant construit le clavier le plus performant dépose une quantité de phéromone supplémentaire sur le système.

III.2.7 Le parallélisme :

Durant chaque itération un nombre prédéfini de fourmis travail en même temps et d'une façon parallèle sur le système. Leur nombre influe de manière relative sur ce dernier, toutefois, les auteurs de la métaheuristique préconisent un nombre de fourmis proche du nombre de villes dans le problème du marchand de commerce.

Dans notre problème le nombre de symboles et de touches est égal à 36, donc, nous avons choisi de fixer le nombre de fourmis à 36.

III.3 Fonctionnement du système

1- Sélection d'un texte à soumettre au système : l'utilisateur choisira un fichier texte de type (txt), à travers l'interface de configuration.

2- Epuration du texte de caractères non traités, et se borner sur le sous-ensemble de caractères qu'on a décidé de traiter.

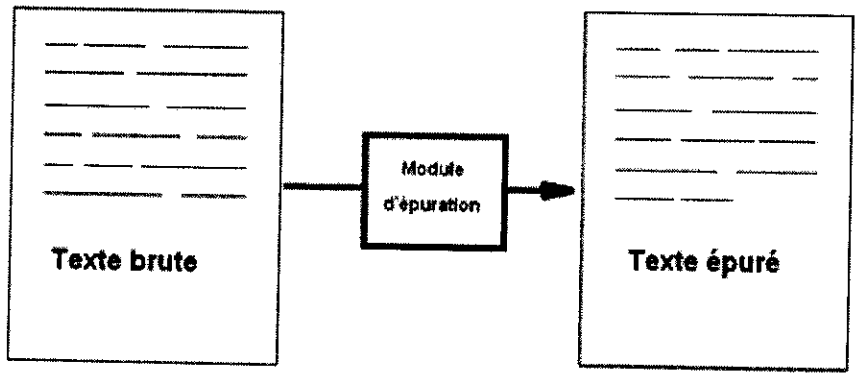


Figure III.6 épuration du texte (Le texte épuré sera exempt de caractères spéciaux, de chiffres et de majuscules).

3- Chaque fourmi recevra une portion de texte de façon à exploité la totalité du texte globale et aussi de faire en sorte que cette portion soit suffisamment grande pour permettre à la majorité des symboles d'apparaitre, des tests seront mené pour déterminer la taille en caractères du texte pour chaque fourmis dans le chapitre suivant (Testes et Résultats).

4- Après la phase d'épuration et de distribution du texte sur les fourmis, chacune d'elle lit le texte qui est en sa possession et place les caractères, un par un, en calculant pour chaque caractère courant C les probabilité $P(C,T)$ qu'il soit placer avec chacune des touche de l'ensemble des touches pas encore allouées dans le clavier partiellement construit par la fourmi K, la touche ayant la probabilité la plus grande sera choisie pour recevoir le caractère C selon la formule (3.1) précédemment citée.

5- à chaque affectation d'un caractère à une touche, les coordonnées de cette dernière, seront gardées en mémoire pour permettre le calcul des distances avec les touches encore vacantes, pour l'affectation du caractère suivant.

Remarque :

A. La première occurrence de chaque caractère dans le texte de chaque fourmi sera la seule à être prise en considération, et à chaque fois qu'on vas ignorer un caractère, parce qu'il figure déjà dans le clavier partiel de la fourmi *k*, la touche ayant le caractère ignoré sera considérer comme la dernière touche allouée, pour permettre un calcul significatif de la visibilité.

B. Les coordonnées de la touche précédemment allouée sur un clavier *Ck* sont essentielles pour calculer les probabilités d'affectation du caractère suivant avec les touches vides, c'est pour cette raison que la première affectation de chaque clavier se fera en ne prenant compte que de la quantité de phéromone (β=0).

C. Ainsi pour la première affectation de la première fourmi de la première itération, les champs de la *matrice des phéromones* n'ont encore subit aucune modification, alors les probabilités seront équivalentes entre elles, c'est pour cette raison que la première affectation sera faite d'une façon aléatoire.

6- Les caractères qui ne sont pas apparues dans la portion du texte alloué à la fourmi *k*, et donc n'ont pas été alloués sur le clavier, seront placé aléatoirement sur les touches encore vacantes de ce dernier *(Ck)*, c'est pour cette raison qu'il faudra attribuer à chaque fourmi une portion de texte suffisamment grande pour minimiser le nombre de caractères aléatoirement attribuées dans les différents claviers construits par les fourmis dans une itération t.

7- Après que chaque fourmi ait construit son propre clavier, la qualité de chacun d'eux sera évaluée selon l'effort fourni par l'utilisateur, cet effort est dans notre cas la distance parcourue par le curseur pour saisir le texte globale *Tg* selon la formule (3.2).

8- Après l'évaluation de chaque clavier d'une itération, l'additivité des phéromones sur la matrice pour chaque clavier sera proportionnel à sa qualité, en calculant : *Q /qk(t,s)*.

Q : est la constante d'additivité.

qk(t,s) :est la qualité du clavier *k* (l'effort) de l'itération t par rapport au texte global *tg*.

La meilleure fourmi déposera plus de phéromone que les autres (élitisme).

9- Le système ne serait pas complet sans le phénomène d'évaporation des phéromones, qui va permettre au système d'oublier les solutions sous-optimales, et d'éviter aussi la saturation de la *matrice de phéromone*, ce processus sera effectué selon la formule suivante :

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) \qquad (3.3)$$

ρ : est le coefficient d'évaporation.

$(1-\rho)$: Quotient de persistance des phéromones.

10- A la fin de chaque itération t, on va comparer le meilleur clavier de cette dernière : « *COT* », avec le clavier optimale globale *COG* et procéder au remplacement éventuel de *COG* par *COT* dans le cas ou ce dernier soit plus performant que *COG*.

III.4 Illustration du processus d'allocation

Dans la partie suivante nous allons illustrer une phase d'affectation d'un caractère par une fourmi *k* dans une itération *t*.

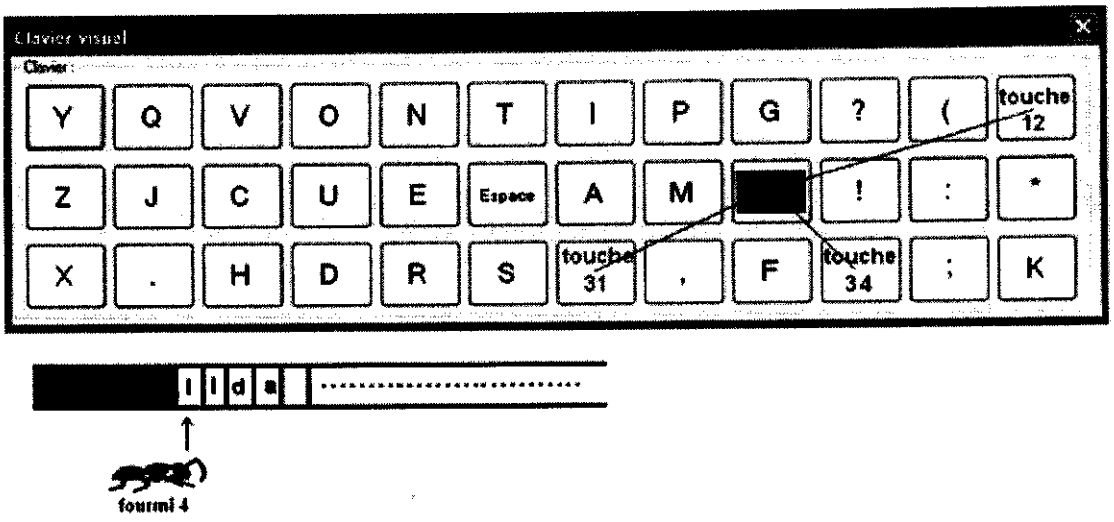


Figure III.7 affectation d'un caractère sur un clavier partiellement construit.

Dans la Figure III.7 la fourmi tente de placer le caractère « *L* », la dernière touche allouée est la touche 21, à laquelle la fourmi a précédemment affecter le symbole « B ». Pour l'affectation suivante (le symbole « *L* »), la fourmis va ,dans un premier temps, vérifier si le symbole existe déjà dans son clavier, et dans le cas ou il en est absent, la fourmi calcule l'inverse de la distance entre la touche précédemment allouée *« touche 21 »* est les touche vacantes *« touche 12, touche 31 et touche 34 ».* Par la suite la fourmi se rapportera à la matrice de phéromone *MP* pour y extraire les quantités de phéromone entre le symbole « L » et chacune des touches vacantes du clavier « 12, 31, 34 ».

	.....	L	.....
.	.....	.	.....
.	.....	.	.....
Touche 12	.....	**2 ,40**	.....
.	.....	.	.....
.	.....	.	.....
Touche 31	.....	**5 ,43**	.....
.	.....	.	.....
.	.....	.	.....
Touche 34	.....	**1 ,32**	.....
.	.....	.	.....

Figure III.8 matrice de phéromone MP

Pour chacun des trois cas, la fourmis k calculera la probabilité *P(C,T)* suivant la formule (3.1), et la touche ayant la probabilité la plus grande recevra le symbole : ***L*** .

Remarque : les distances sont calculées en pixels.

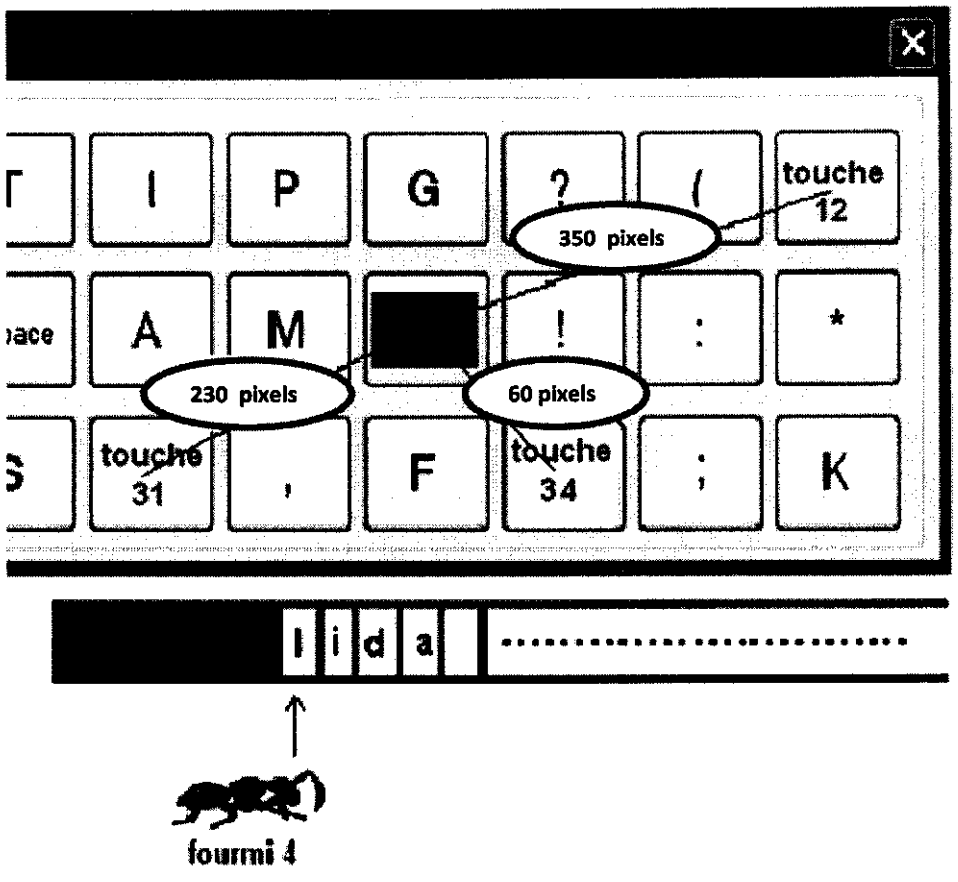


Figure III.9 Distance entre les touches.

Distance entres les touche :	Quantité de phéromones
$d\varepsilon$ (t21, t12) = 350 pixels	$\tau(l, t12) = 2,40$
$d\varepsilon$ (t21, t31) = 230 pixels	$\tau(l, t31) = 5,43$
$d\varepsilon$ (t21, t34) = 60 pixels	$\tau(l, t34) = 1,32$

- $\alpha = 3$
- $\beta = 2$

Calcule des probabilités d'allocation :

$$P_K(\text{« L »}, t12) = \frac{2{,}40^3 * 350^2}{(2{,}40^3 * 350^2)+(5{,}43^3 * 230^2)+(1{,}32^3 * 60^2)} = 0{,}166494$$

$$P_K(\text{« L »}, t31) = \frac{5{,}43^3 * 230^2}{(2{,}40^3 * 350^2)+(5{,}43^3 * 230^2)+(1{,}32^3 * 60^2)} = 0{,}832691$$

$$P_K(\text{« L »}, t34) = \frac{1{,}32^3 * 60^2}{(2{,}40^3 * 350^2)+(5{,}43^3 * 230^2)+(1{,}32^3 * 60^2)} = 0{,}000814$$

$P_K(\text{« L »}, t31)$ est la probabilité la plus grande, donc, la fourmi *k* placera le symbole «L» sur la touche 31.

III.4.1 Choix du nombre d'itération

La condition d'arrêt du system est fixée au départ, par le choix d'un nombre d'itérations à accomplir. La désignation d'un nombre élevé d'itérations est à préconiser à cause de l'inexistence d'une garantie de convergence dans un system basé sur une métaheuristique de colonie de fourmis comme c'est le cas dans notre travail.

L'utilisateur pourra stopper le processus manuellement en cas de convergence, ou de fluctuation continuel, ce dispositif a été envisagé suivant le réglage des paramètres dans la phase d'initialisation, la convergence (si elle aura lieu) sera rapide ou lente, c'est pour cette raison qu'un nombre d'itération élevé sera à prévoir pour satisfaire les deux cas.

III.4.2 Algorithme de fonctionnement du système

Choisir le nombre d'itération ***T*** *; Choisir le nombre de fourmis* ***N*** *;*

Designer ***Nt, Nc*** *: le nombre de caractères et de touches; // dans notre cas* ***Nt=Nc***

Designer ***L*** *= la longueur du texte source;*

Choisir un quotient d'additivité des phéromones ***Q;***

Choisir un quotient de persistance des phéromones ***P*** *avec* ***0 < P < 1;***

Initialiser la matrice de phéromones ***Mp[Nt][Nc]*** *avec une quantité différente de* ***0;***

Initialisation

Pour *j :=* ***1*** *jusqu'à* ***T*** *faire*

Créer ***N*** *texte :* ***S[N][L]*** *;*

Créer ***N*** *clavier vide* ***T[N][Nt]*** *;*

Pour *i :=* ***1*** *jusqu'à* ***L*** *faire*

Pour *k :=* ***1*** *jusqu'à* ***N*** *faire*

C := S[k][i] *;*

Si *la lettre* ***C*** *n'a pas été traitée par la fourmi* ***k*** *faire*

Choisir une position ***P*** *pour la lettre* ***C*** *dans le clavier partiellement construit par la fourmi K* ***T[k][p] := C*** *selon la formule (3.1)*

Fin Si

Fin pour

Fin pour

Affecter aléatoirement les caractères manquant sur les touche encore vacantes de chaque clavier de l'itération ***j*** *;*

Pour *k :=* ***1*** *jusqu'à* ***N*** *faire*

Calculer ***Vk*** *l'évaluation du clavier construit par la fourmi* ***k*** *; selon la formule (3.2)*

Ajouter une quantité de phéromone a la matrice MP , proportionnelle a la qualité du clavier construit par la formis ***k***

MP[k] $\longleftarrow$ $\frac{Q}{Vk}$;

Fin pour

Evaporer l'ensemble des pistes de phéromone suivant la formule (3.3)

Fin pour

Itération

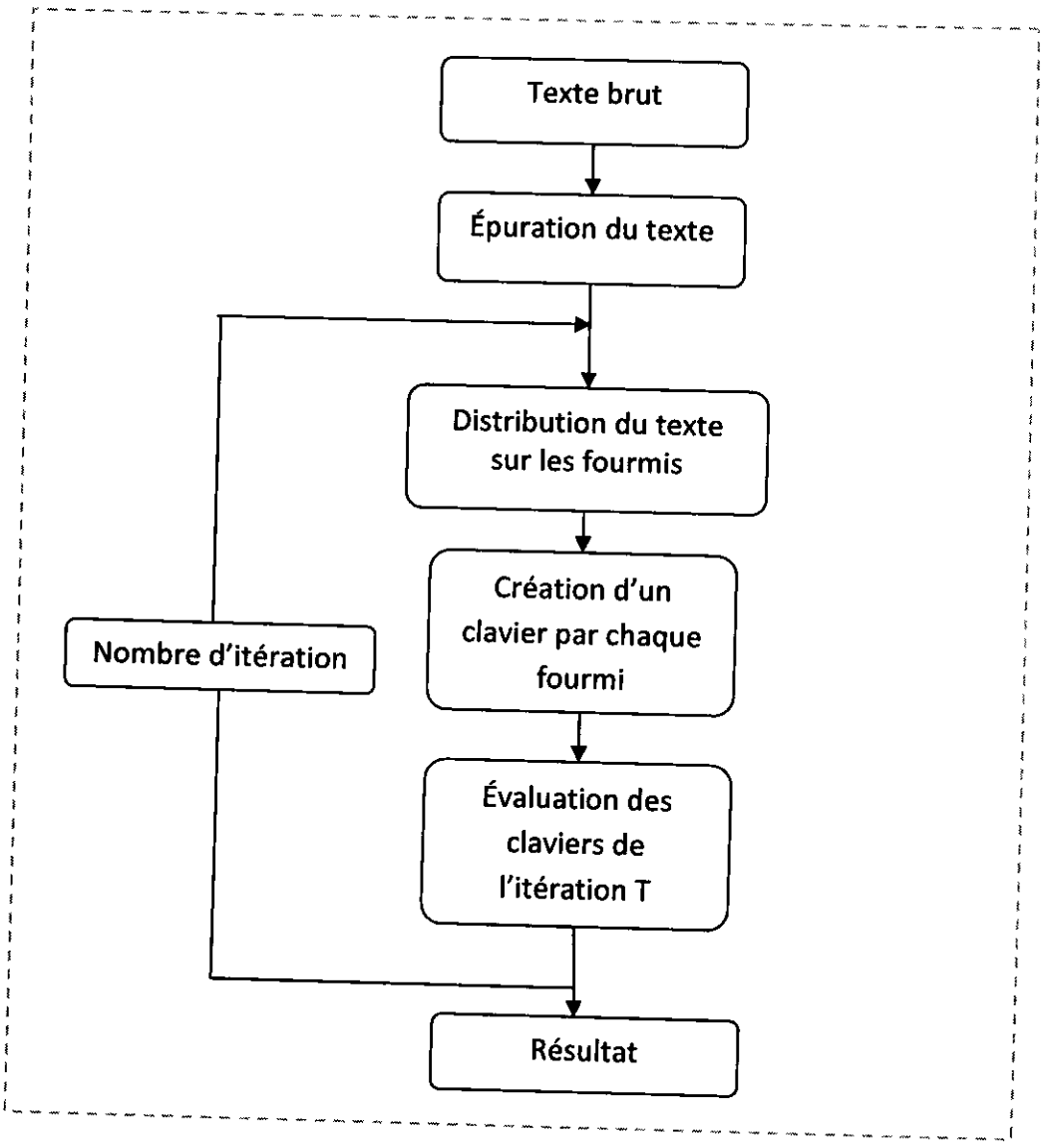


Figure III.10 Déroulement du système.

III.5 Mise en œuvre

III.5.1 Environnement de mise en œuvre :

Pour implémenter notre méthode on a utilisé le langage *C#* et *Microsoft Visual .net 2008* comme un environnement de développement intégré sous une plateforme *Windows* XP ou Vista.

On a choisi cette combinaison pour sa gestion automatique très efficace de la mémoire, pour sa prise en main assez facile et pour sa palette très large d'outils de création.

On a utilisé aussi *DXperience* (un outil graphique) pour créer les différents graphes utilisés pour le suivi et l'interprétation des résultats.

III.6 Interface de l'application

Notre application regroupe 4 interfaces :

- Interface principale.
- Interface de configuration.
- Interface des graphes.
- Clavier optimale.

III.6.1 Interface principale

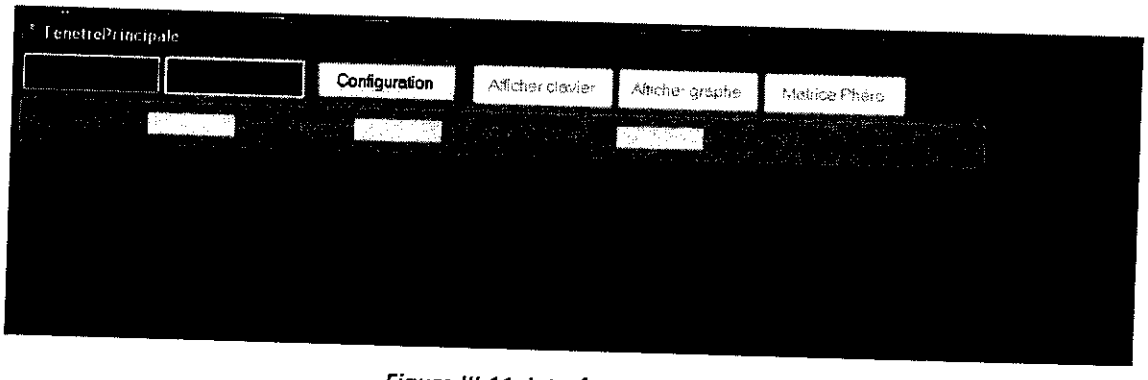


Figure III.11 interface principale.

L'interface principale inclue l'affichage de l'évaluation du meilleur clavier globale, ainsi que l'évaluation du meilleur clavier de l'itération en cours, elle inclue aussi des

boutons permettant d'accéder au autres interfaces (configuration, graphes et clavier).

L'utilisateur peut modifier la configuration des paramètres en appuyant sur le bouton configuration.

III.6.2 L'interface de configuration des paramètres

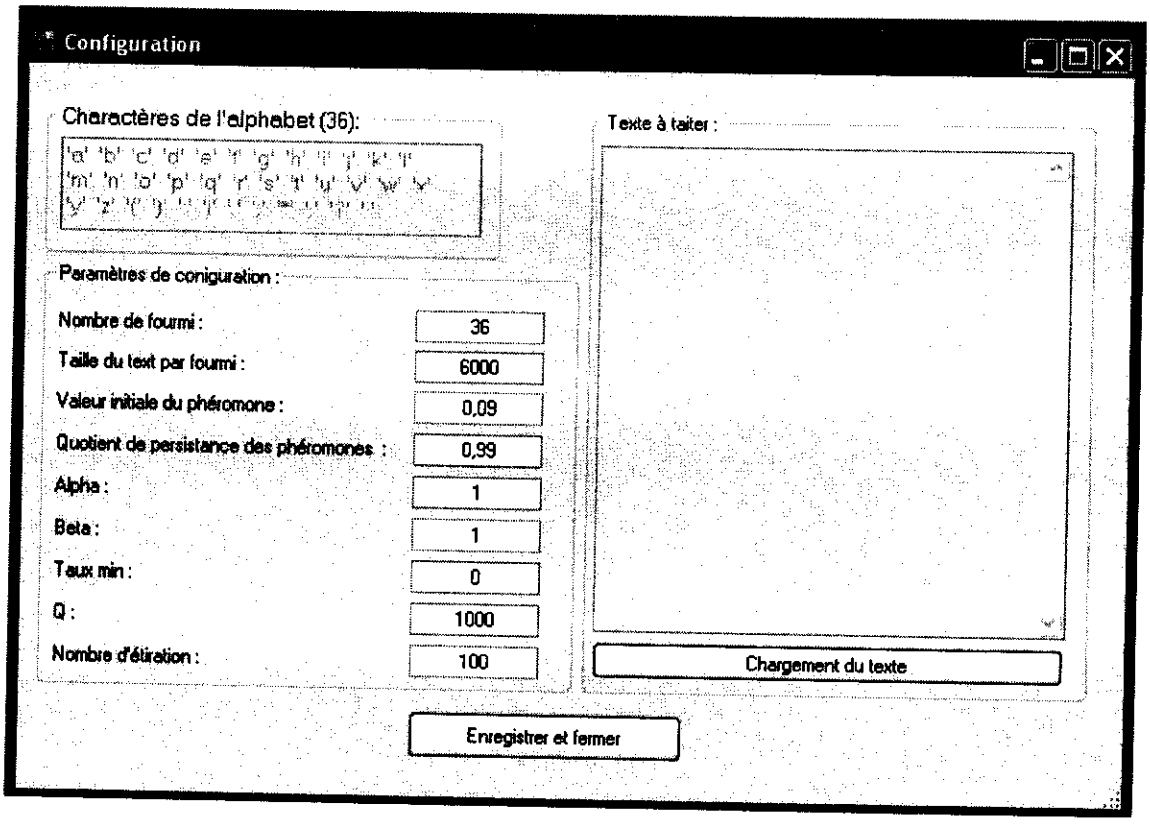


Figure III.12 interface de configuration.

L'interface de configuration permet à l'utilisateur de choisir les paramètres du système : (α, β, p, nb de fourmis ..... etc.).

III.6.3 interface des graphes

Cette interface permet de voir l'évolution du système au cours des itérations.

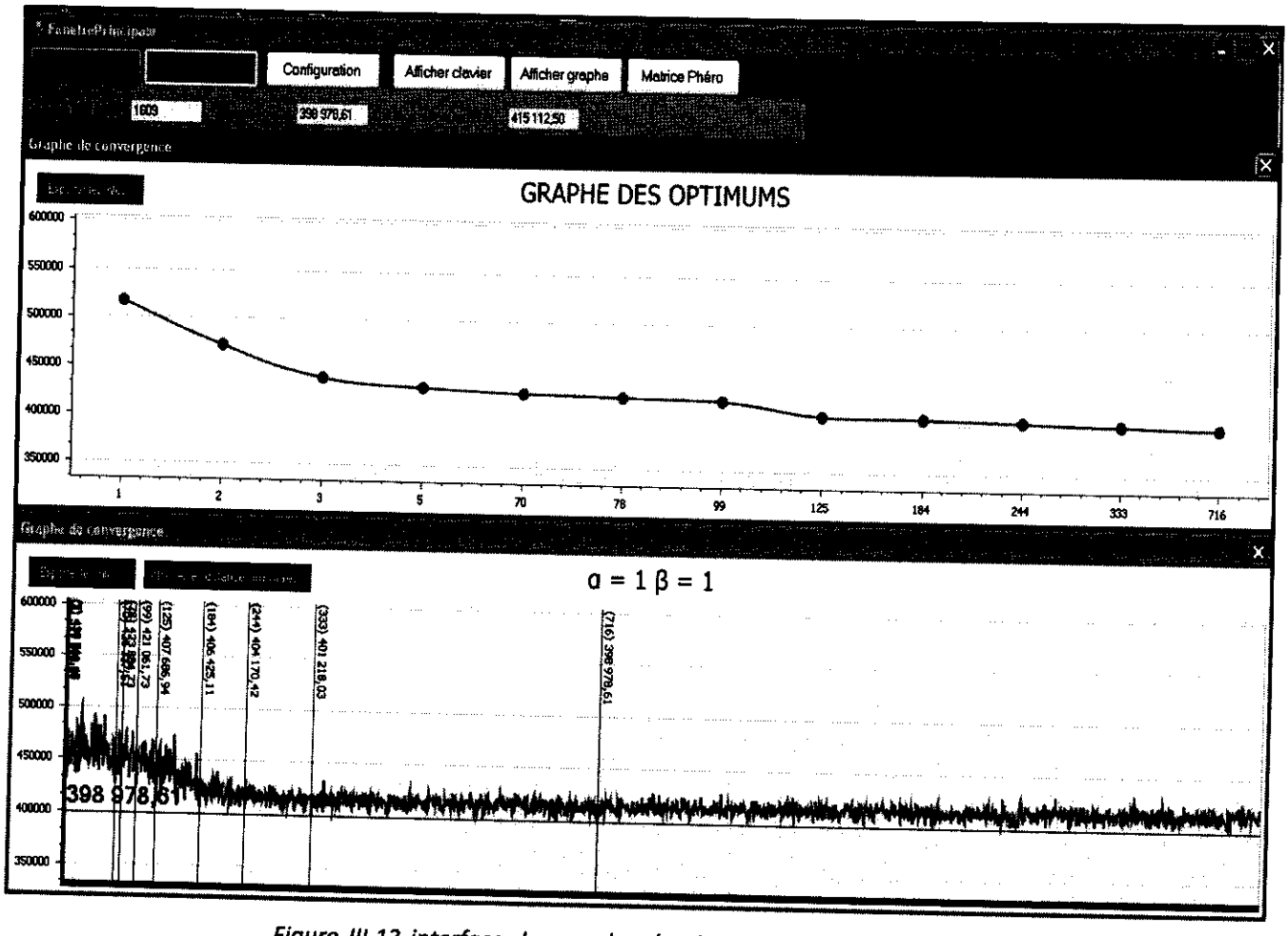


Figure III.13 interface des graphes (en haut : graphe des optimums, en bas : graphe de convergence)

- Le graphe de convergence permet de voir l'évolution des évaluations des claviers successifs.
- le graphe des optimums permet de voir l'évaluation de chaque clavier optimale par rapport à l'itération sur laquelle il est apparu.

III.6.4 Clavier optimal

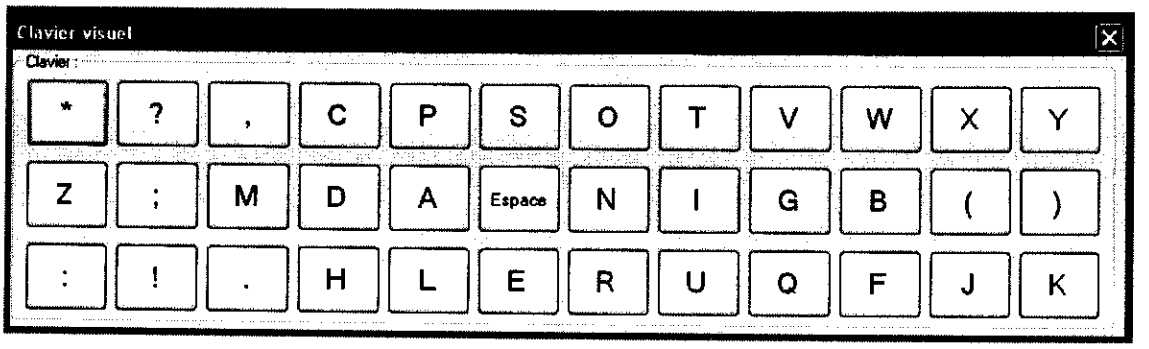


Figure III.14 interface clavier.

Cette interface permet d'afficher le clavier optimal obtenu au cours des itérations précédentes.

III.7 Conclusion

Comme guise de conclusion nous pouvons dire que ce présent chapitre représente la concrétisation de l'étude théorique qui a été mené en amont.

Jusqu'à présent nous avons mis en place un algorithme d'optimisation et nous l'avons implémenté dans un programme informatique, toutefois, un nombre donné de paramètres influe sur ce dernier (α, β, p, Q ..... etc.), dans le chapitre qui suit, nous tenteront d'évaluer l'influence de chacun d'eux, et par la suite, désigner une configuration optimale de ces paramètres.

Chapitre IV

TESTS ET RESULTATS

Chapitre IV : TESTES ET RESULTATS

Dans ce chapitre nous allons procéder à une série de tests en variant les différents paramètres du système, puis dans un deuxième temps analyser et comparer les résultats obtenus.

Matériels utilisés

Les testes ont été effectué sur un Pc doté d'un processeur Intel® Pentium® Dual-Core™ 1.6 GHz avec 2 Go de RAM sur une plateforme Windows XP & Vista.

IV.1. Etude comparative

IV.1.1. Choix des paramètres

Les différents paramètres du système ont une influence relative sur son comportement global, nous tenterons d'évaluer cette influence pour chacun d'eux, afin d'obtenir des résultats satisfaisant.

Les paramètres étudiés sont :

- *N* : le nombre de fourmis pour chaque itération.
- α, β : les paramètres contrôlant l'importance relative de l'intensité de la piste et de la visibilité.
- *P* : persistance de la trainée qui détermine la quantité de phéromones restantes après chaque phase d'évaporation.
- *Q* : quotient de l'additivité des phéromones.
- *Lt*: taille du texte attribué à chaque fourmi.
- *T* : nombre d'itérations.

IV.1.2. Choix du nombre de fourmis N et du quotient d'additivité des phéromones Q

Les auteurs de la métaheuristique des colonies de fourmis préconise un nombre de fourmis proche du nombre de villes dans un problème de voyageur de commerce, en ce basant sur ces recommandation, nous avons choisie de faire nos testes avec un nombre de fourmi égale à 36, correspondent au nombre de caractères dans notre clavier, En ce qui concerne le quotient d'additivité les auteurs disent qu'il n'influe que d'une façon minimale sur le résultat obtenu ce qui a été confirmé durant nos tests.

IV.1.3. Choix du nombre de caractères attribué par chaque fourmi *Lt*

Le nombre de caractères attribué à chaque fourmi joue un rôle important dans le déroulement du système, un nombre de caractères élevé est idéal pour diminuer les affectations aléatoires des caractères non apparu dans le segment de texte attribué à chaque fourmis, mais ralenti son déroulement. En revanche, un nombre peu élevé de caractères par fourmi accélérera le déroulement du processus au dépend de l'augmentation du nombre de symbole aléatoirement attribué.

Nous avons effectué quelque testes, pour trouver un compromis entre les deux cas et designer un nombre idéal de caractères par fourmis. Ces testes sont été mené de la manière suivante :

On varie le nombre de caractère par fourmi et on calcule la moyenne des symboles aléatoirement alloués pour une itération.

Les figures suivantes sont la représentation mémoire des claviers partiellement construits par chaque fourmi avant l'allocation aléatoire des symboles non apparus dans les portions du texte alloué à chacune d'elle, chaque ligne *i* est la représentation linéaire du clavier construit par la fourmis *k* et les caractères « \0 » sont les touche vacantes de ce dernier.

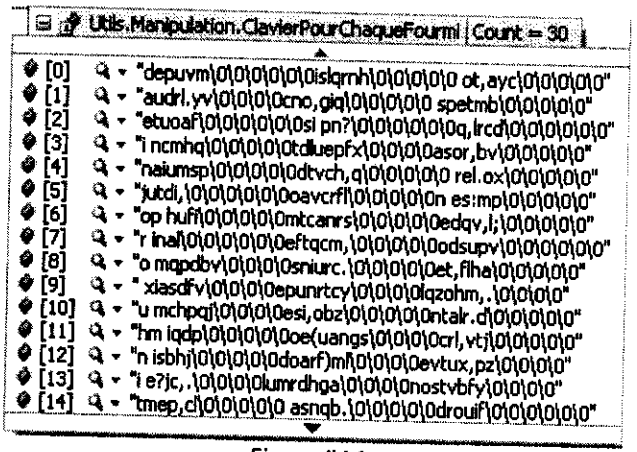


Figure IV.1

100 Caractères attribués pour chaque fourmi.

13 ,26 symboles sur 36 attribué aléatoirement en moyenne.

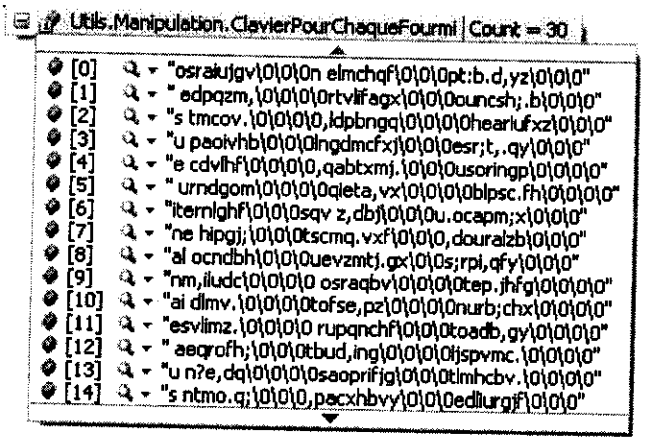


Figure IV.2

400 Caractères attribués pour chaque fourmi.

10,33 symboles sur 36 attribués aléatoirement en moyenne.

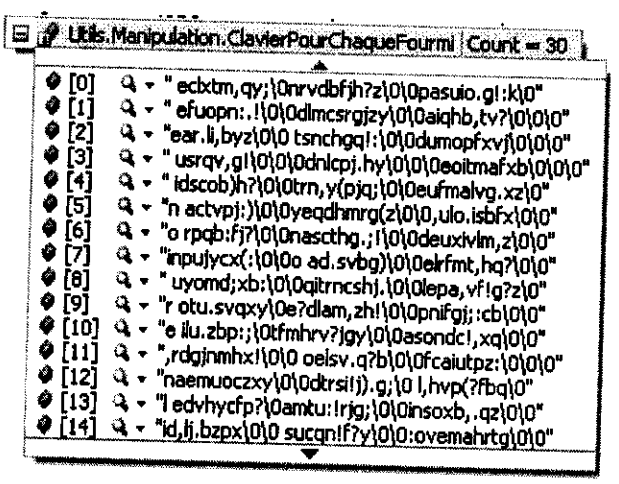


Figure IV.3

1000 Caractères attribués pour chaque fourmi.

5,86 symboles attribués aléatoirement en moyenne.

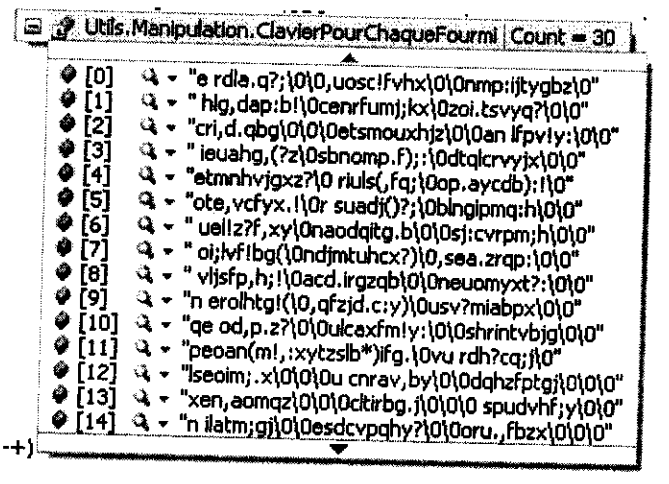


Figure IV.4

1700 Caractères attribués pour chaque fourmi.

5 symboles attribués aléatoirement en moyenne.

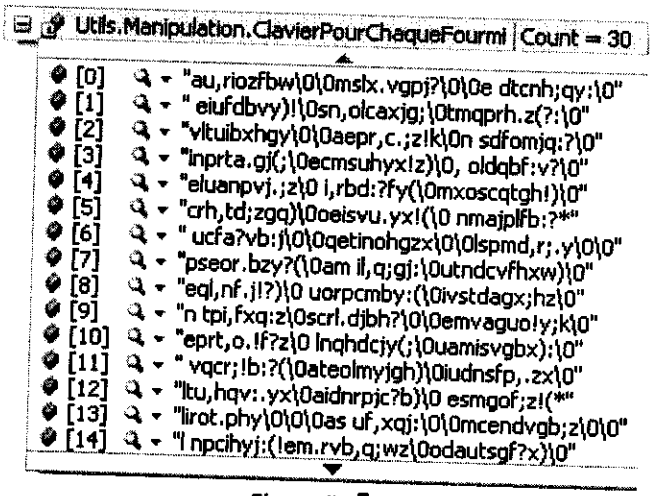


Figure IV.5

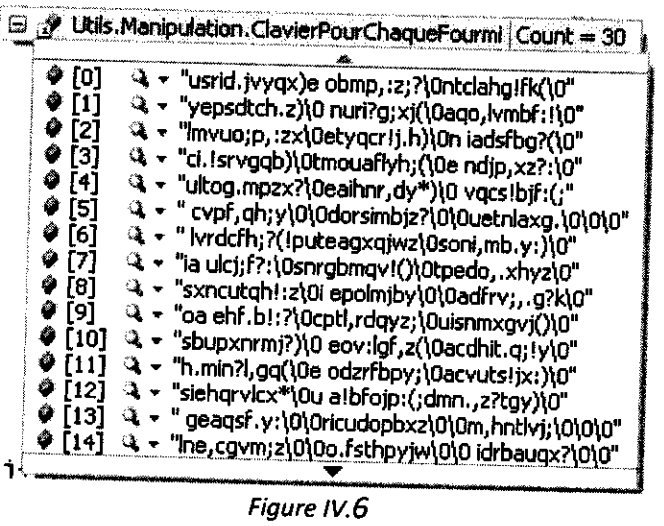


Figure IV.6

4000 Caractères attribués pour chaque fourmi.

3,4 symboles attribués aléatoirement en moyenne.

5000 Caractères attribués pour chaque fourmi.

3,26 symboles attribués aléatoirement en moyenne.

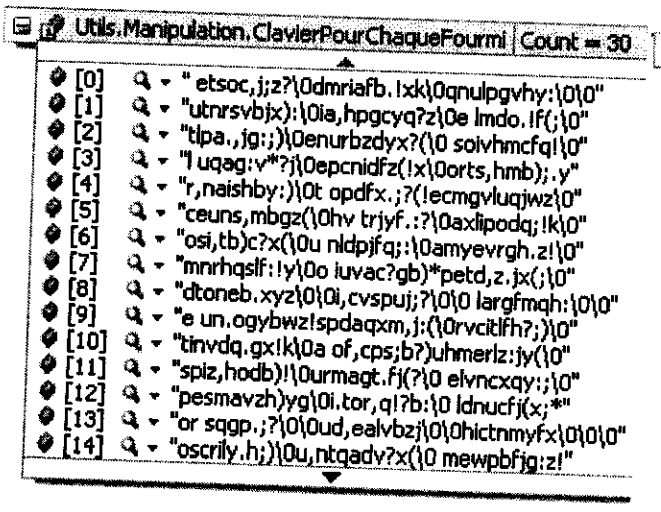


Figure IV.7

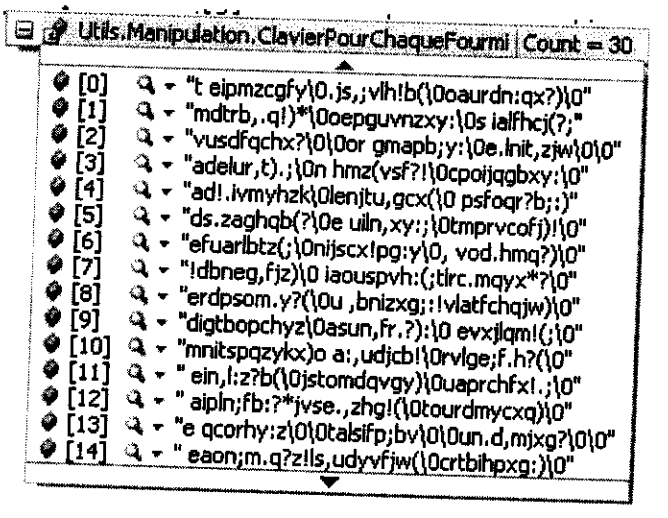


Figure IV.8

6000 Caractères attribués pour chaque fourmi.

3 symboles attribués aléatoirement en moyenne.

7000 Caractères attribués pour chaque fourmi.

2,8 symboles attribués aléatoirement en moyenne.

Le graphe suivant montre la variation entre le nombre de caractères alloués à chaque fourmi et le nombre de symboles attribués aléatoirement.

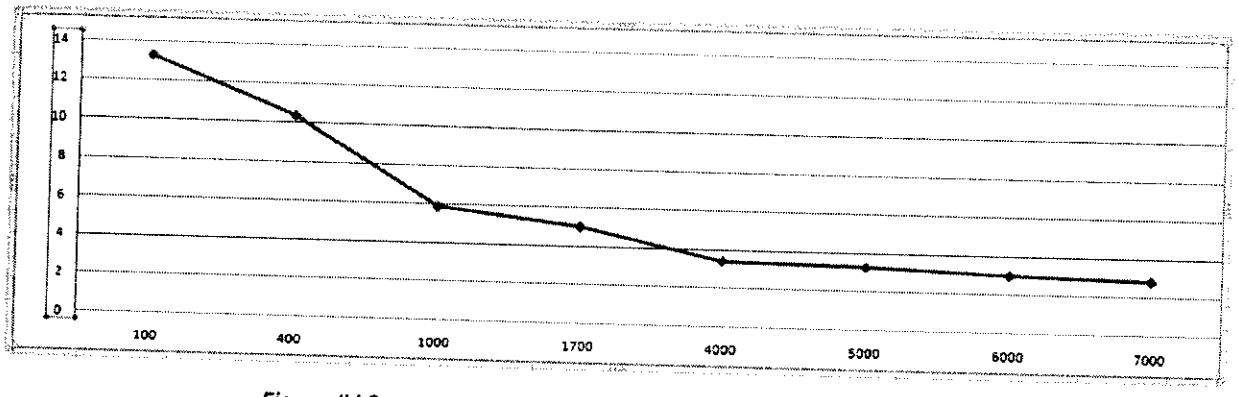


Figure IV.9 moyennes des symboles attribués aléatoirement.

Après avoir fait les tests précédents, nous avons constaté qu'au-delà de 6000 caractères par fourmis, les symboles aléatoirement alloués sont au voisinage de 3 à chaque fois, donc les tests que nous allons effectuer pour l'optimisation se feront avec un nombre de caractères par fourmis *Lt* =6000.

Une plus grande taille ne sera d'aucune utilité expérimentale, en plus de l'*augmentation significative* du temps de calcul qu'elle va engendrer.

IV.2. Condition d'expérimentation

- *Lt*=6000 caractères : taille du texte pour chaque fourmi.
- *N*=36 fourmis
- *Q*=1000

Remarque :

Sur l'ensemble des testes que nous avons mené, nous avons utilisé un texte unique contenant 35 000 caractères, de type littéraire français.

L'unicité du texte est essentielle pour permettre une comparaison significative entre les tests.

La taille de ce texte a été choisie en respectant un compromis entre une taille trop grande qui alourdirai le déroulement du système et augmenterai significativement le temps

d'exécution des tests et une taille trop petite qui aura pour conséquence une représentation pas assez significative des caractères et des combinaisons de caractères.

1er test :

$\alpha = \beta$

Posons $\alpha = 1$ et $\beta = 1$.

- *P*=0,99

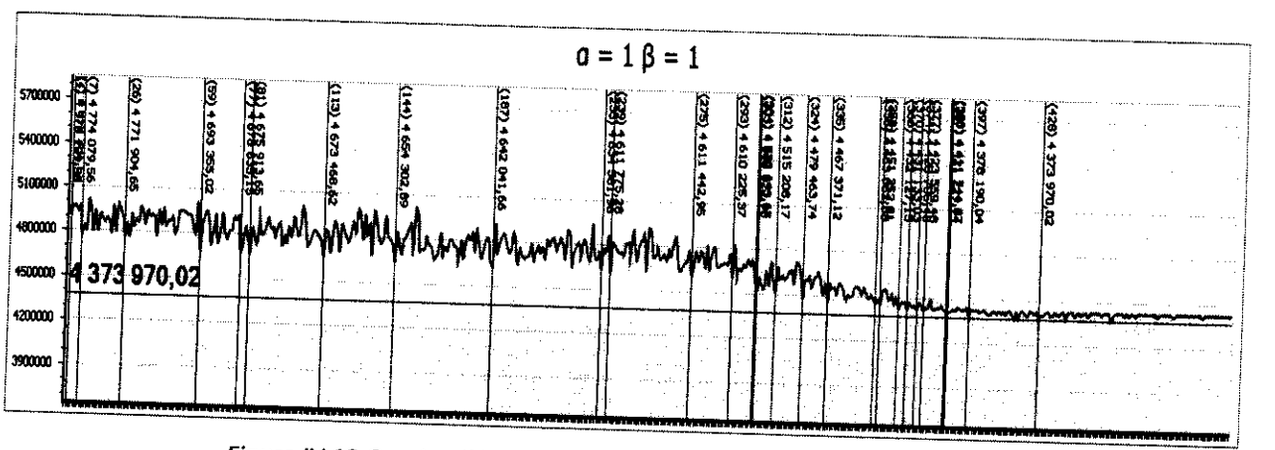


Figure IV.10 Graphe d'évaluation de l'effort généré pour saisir le texte globale Tg sur le meilleur clavier de chaque itération.

Début de la convergence à partir de la 250ème itération avec la valeur : 4 373 970,02 pixels.

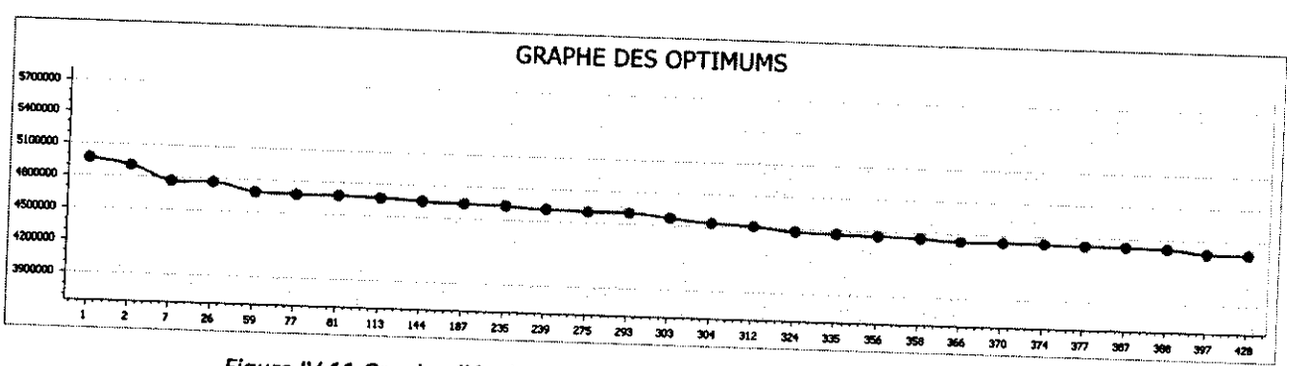


Figure IV.11 Graphe d'évaluations des claviers optimales successif.

Au début on remarque une fluctuation dû à la diversification des solutions, puis une convergence à partir de l'itération 275 pour aboutir au résultat finale a l'itération 420, la

convergence a eu lieu à cause de la concentration des phéromones sur les combinaison (touche, symbole) les plus fréquemment associées par les fourmis.

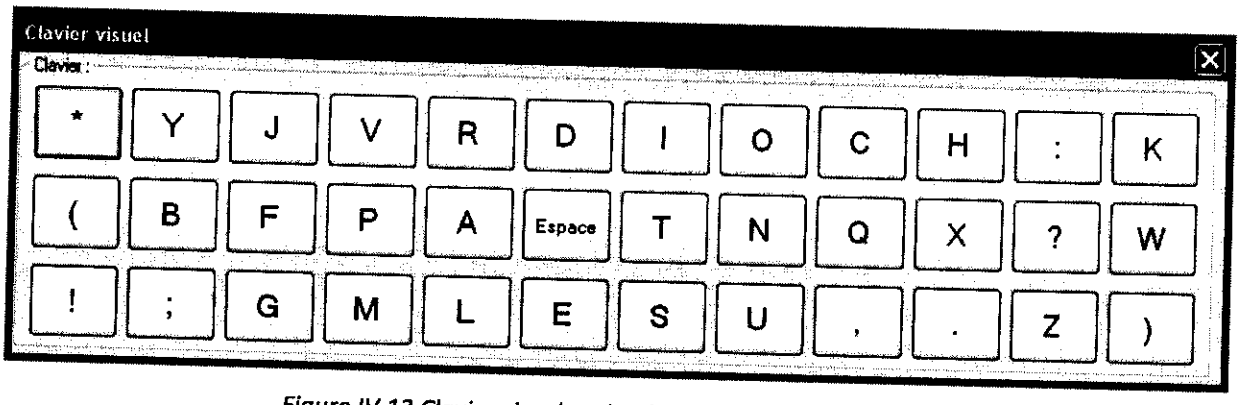


Figure IV.12 Clavier visuel optimal après convergence du système.

- pour P=0,7 (et que α et β sont toujours à 1)

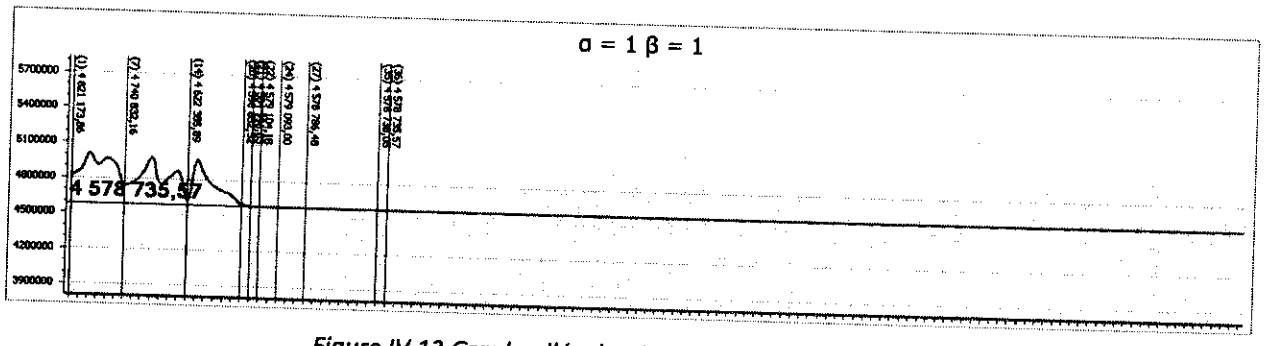


Figure IV.13 Graphe d'évaluation de l'effort généré.

Après quelques itérations une convergence prématurée a eu lieu, ceci est dû à une trop grande évaporation des phéromones qui a conduit à l'oubli par le système d'un grand nombre de solutions prometteuses.

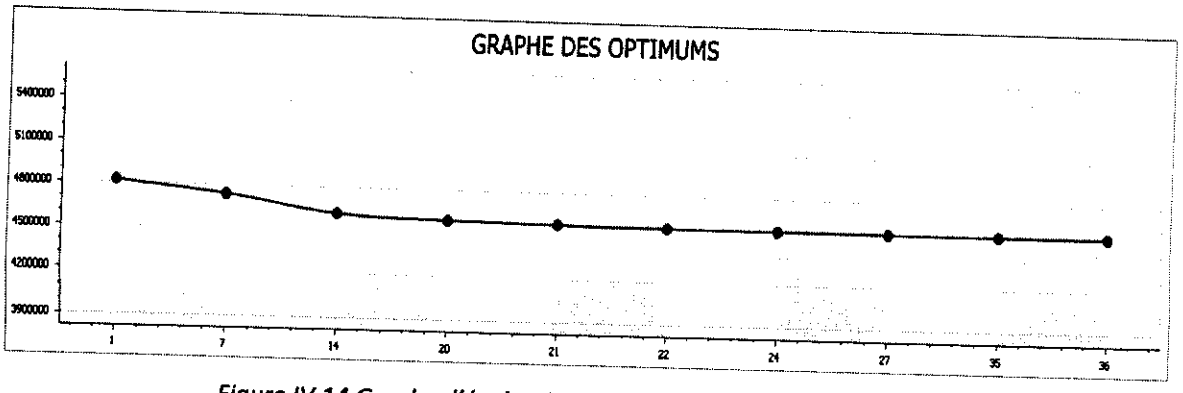


Figure IV.14 Graphe d'évaluations des claviers optimales successif.

- *P*=0,5 (α =1 β=1)

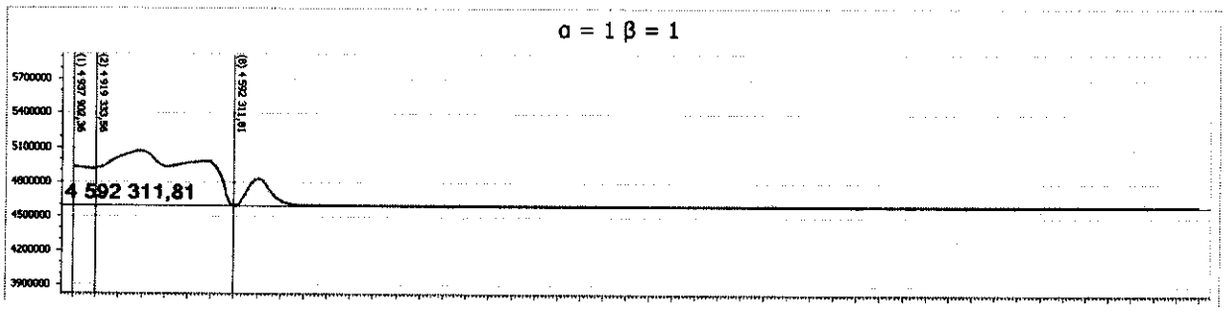


Figure IV.15 Graphe d'évaluation de l'effort.

Une trop grande évaporation a fait que les premières pistes moins fournie en phéromones ont perdus leurs attractivité prématurément par rapport au pistes plus phéromonés au départ, de la même façon que sa c'est passé dans le test précédent, et une convergence a eu lieu dès les premières itérations.

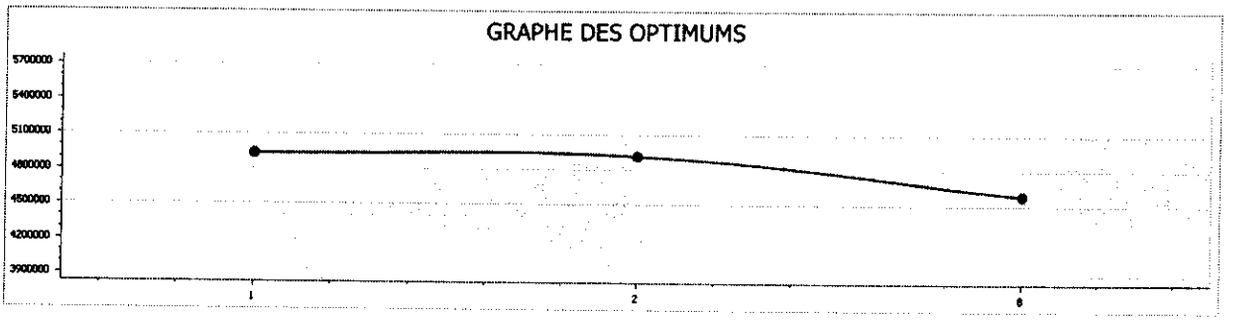


Figure IV.16 Graphe d'évaluations des claviers optimales successif.

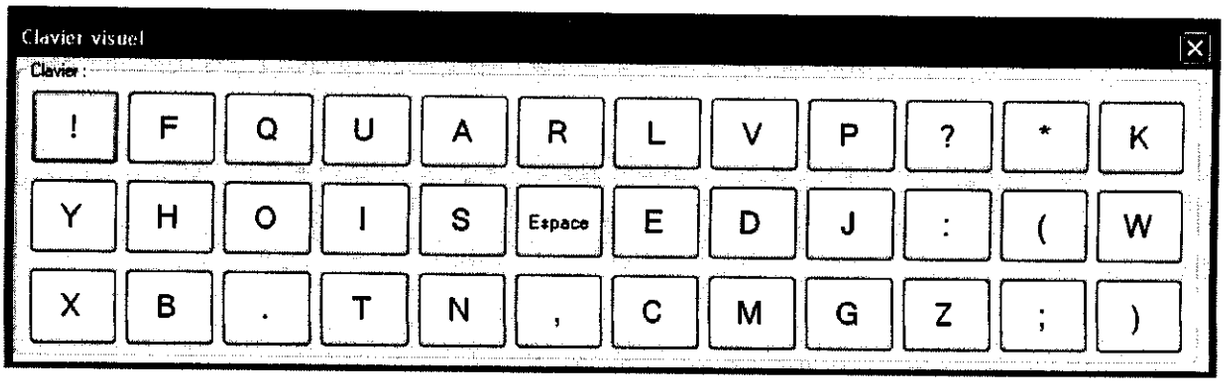


Figure IV.17 Clavier visuel optimal après convergence du système

- P=0,7 (α =2 β=1)

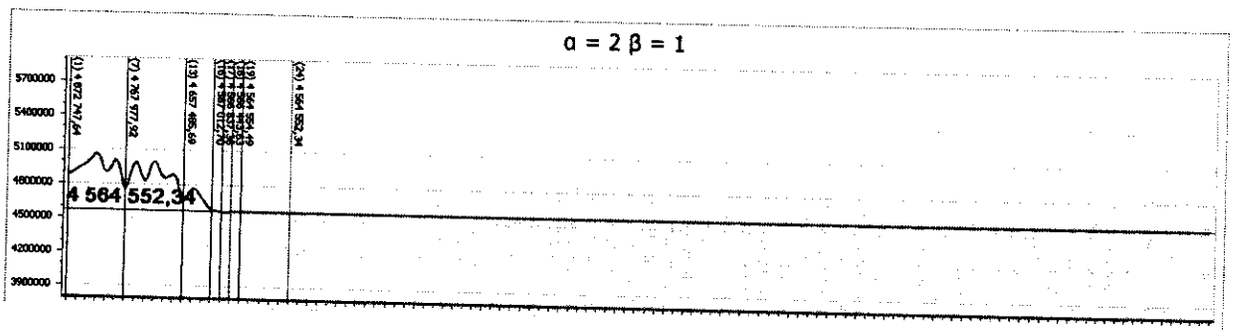


Figure IV.21 Graphe d'évaluation de l'effort.

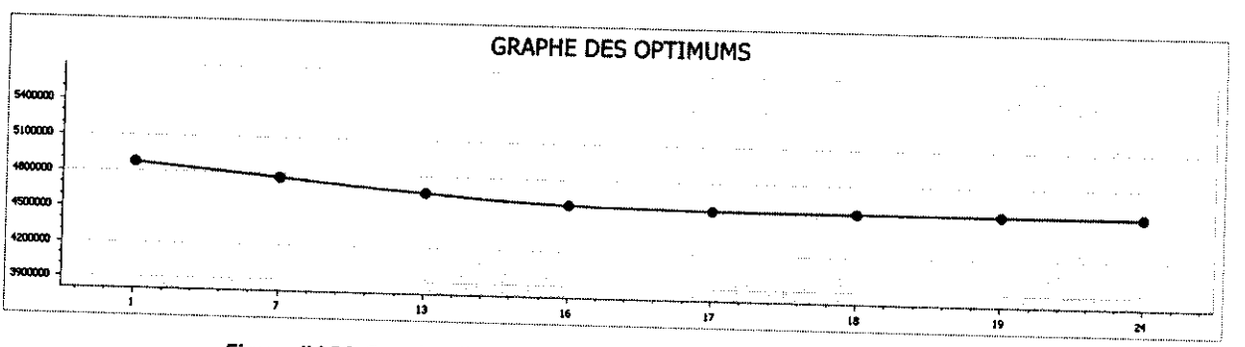


Figure IV.22 Graphe d'évaluations des claviers optimales successif.

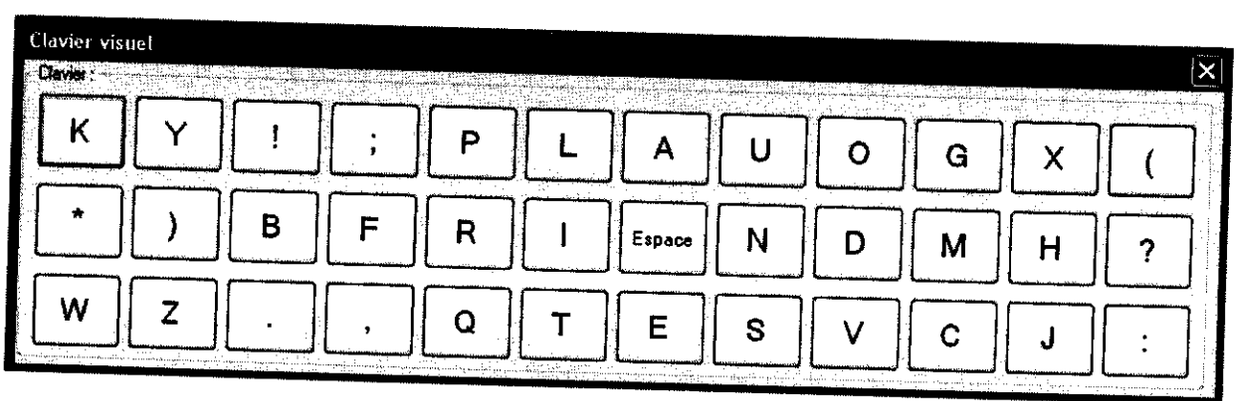


Figure IV.23 Clavier visuel optimal après convergence du système

- P=0,5 (α =2 β=1)

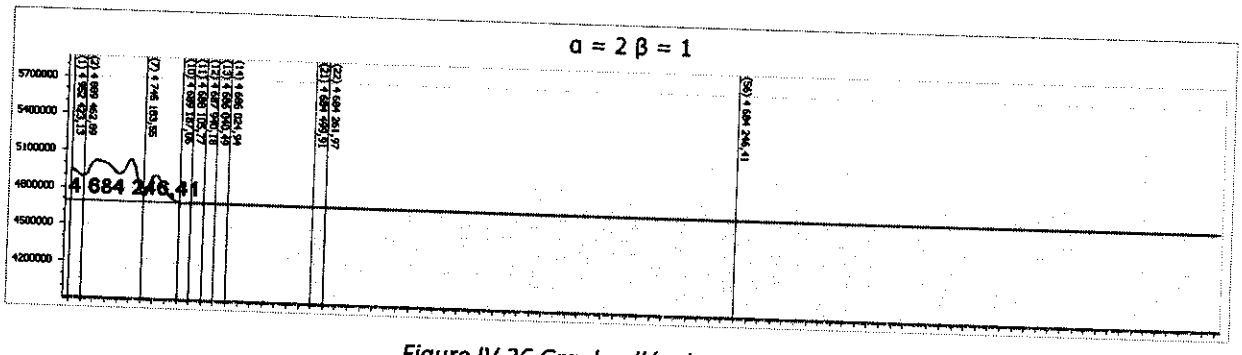


Figure IV.26 Graphe d'évaluation de l'effort.

Une trop grande évaporation a éliminé un grand nombre de solutions envisageable et à conduit une convergence prématuré du système à l'itération 10.

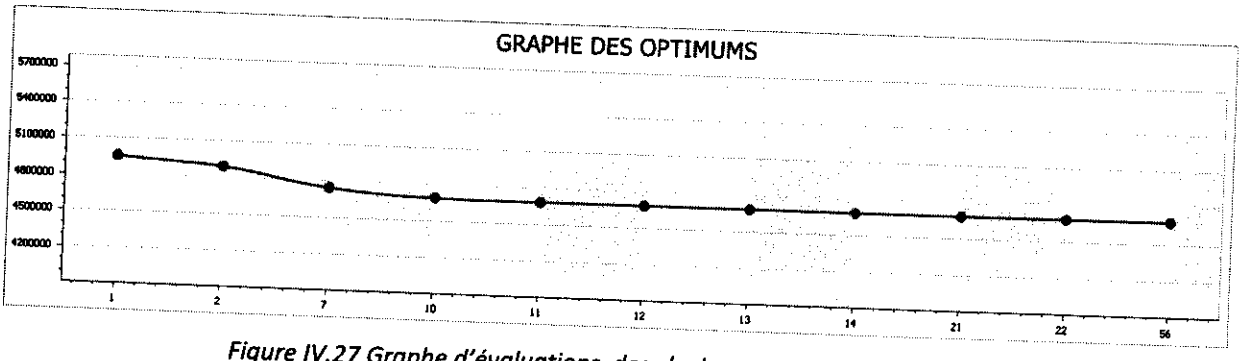


Figure IV.27 Graphe d'évaluations des claviers optimales successif

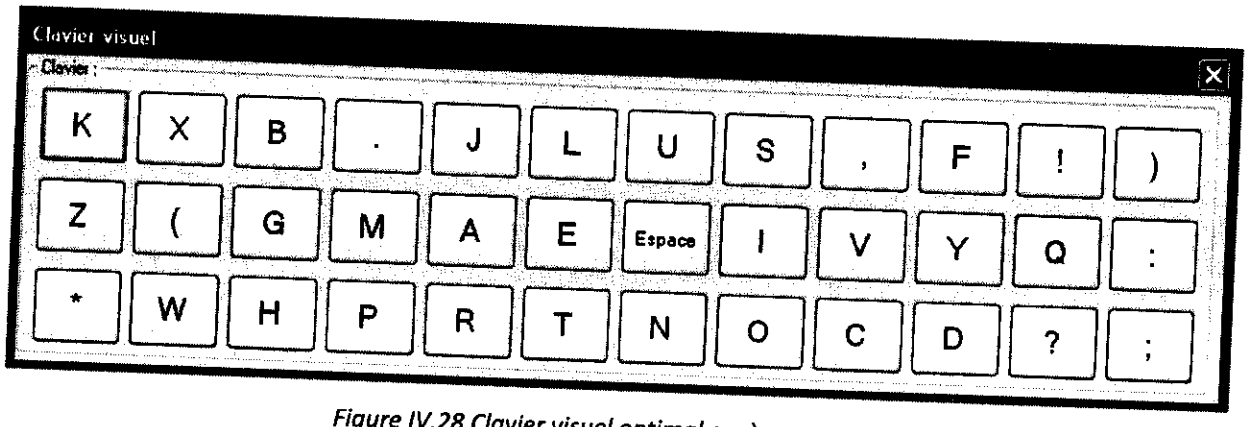


Figure IV.28 Clavier visuel optimal après convergence du système

3ème test :

α =1 β=0 (On a éliminé la visibilité, ne laissant que l'attractivité des pistes des phéromones)

- p=0,99

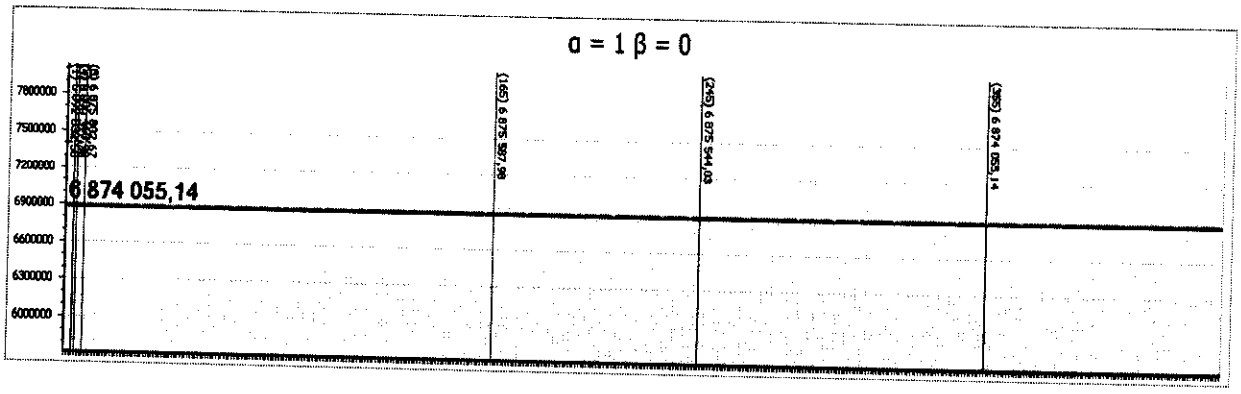


Figure IV.29 Graphe d'évaluation de l'effort.

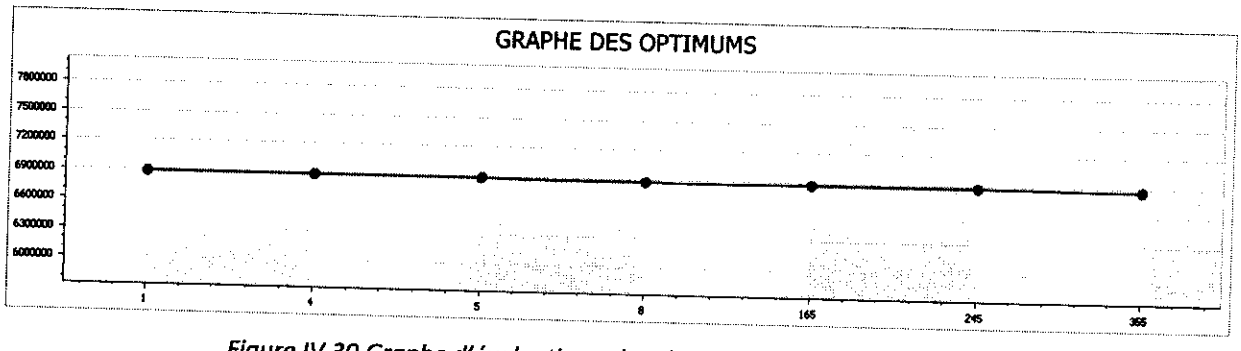


Figure IV.30 Graphe d'évaluations des claviers optimales successif.

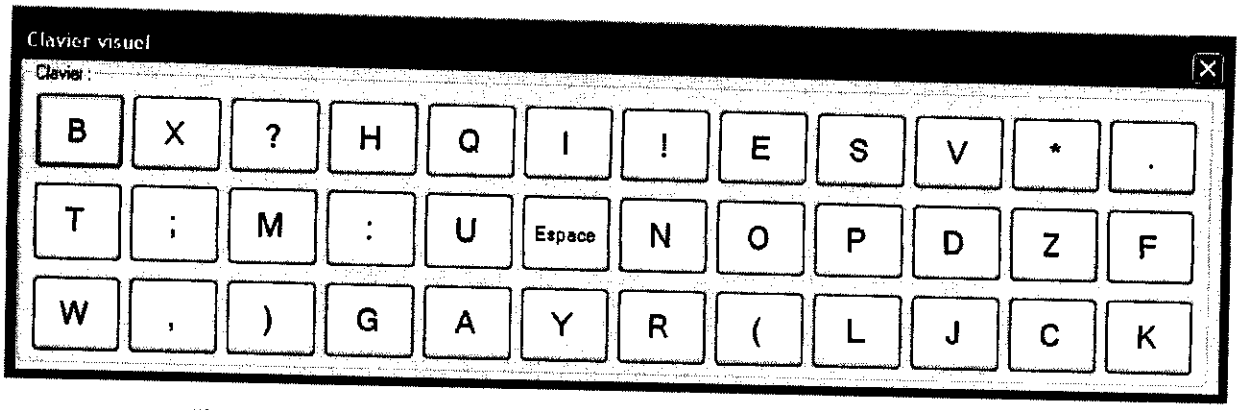


Figure IV.31 Clavier visuel optimal après convergence du système.

En enlèvent la visibilité (β=0) les fourmis n'ayant aucune indications sur la distance, chacune d'elle n'a suivi que l'attractivité des pistes des phéromones en accentuant à chaque fois cette dernière, ce qui a fait que le système a convergé dès le début au voisinage du premier résultat.

4ème test :

α =5 β=1

- *P*=0,99

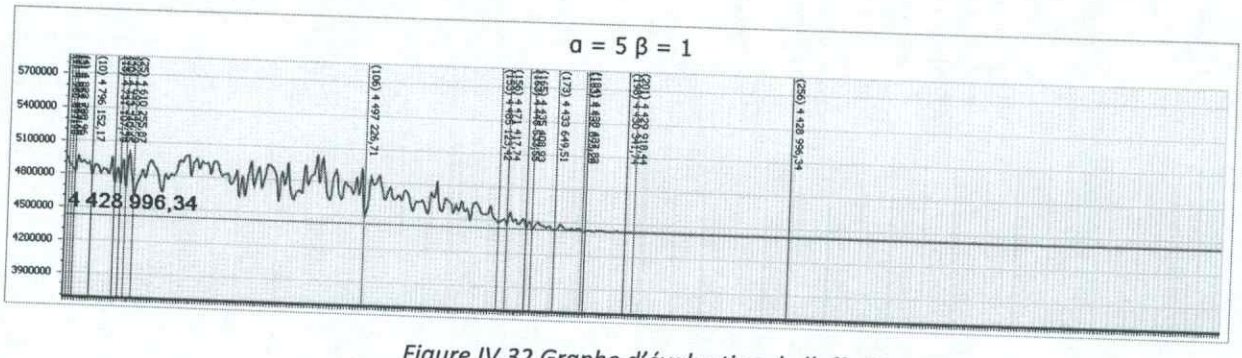


Figure IV.32 Graphe d'évaluation de l'effort.

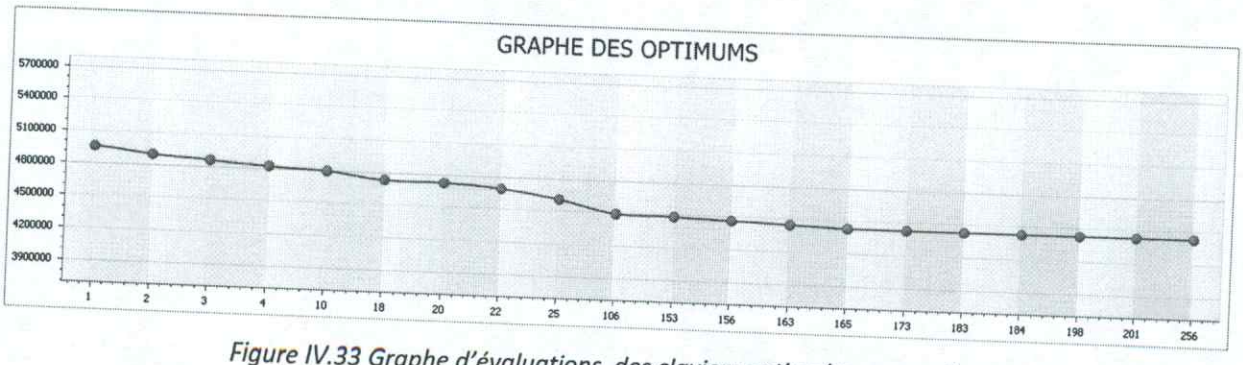


Figure IV.33 Graphe d'évaluations des claviers optimales successif

Début de la Convergence après 100 itérations et une convergence totale aux alentours de la 200ème itération avec le clavier suivant :

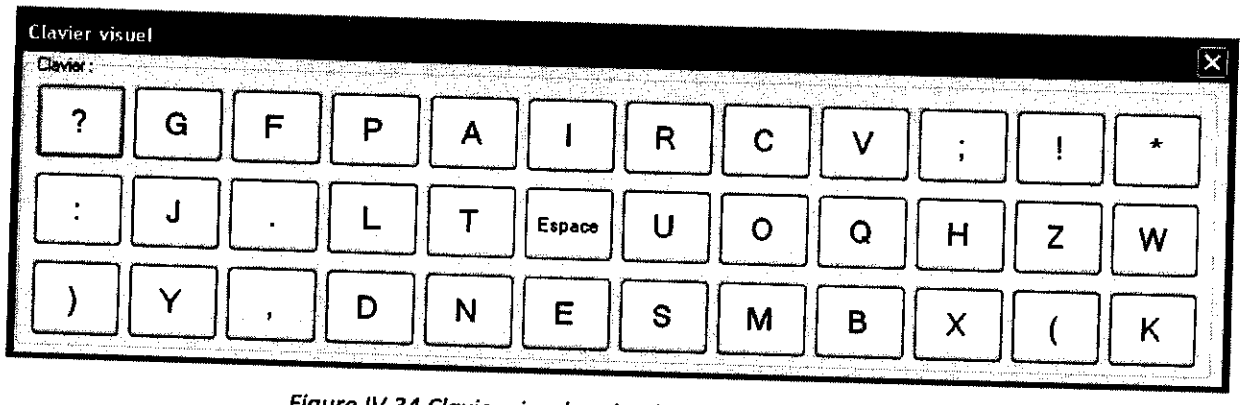


Figure IV.34 Clavier visuel optimal après convergence du système

- P=0,7 (α =5 β=1)

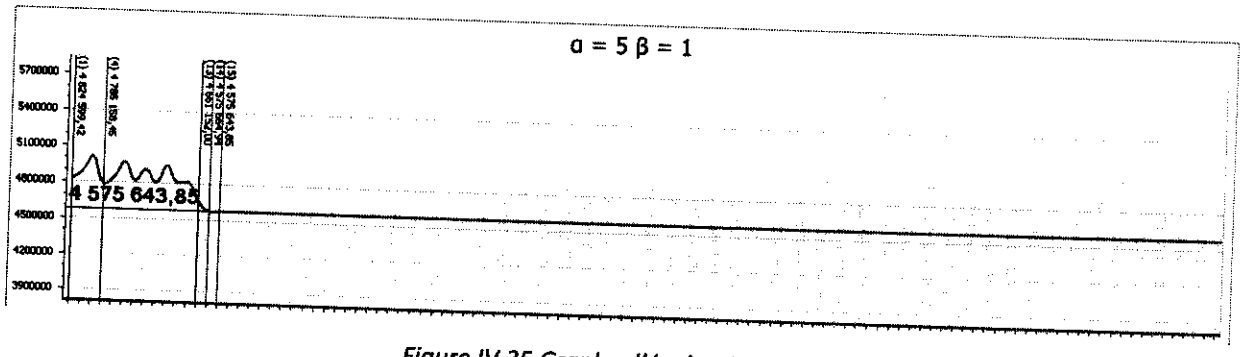


Figure IV.35 Graphe d'évaluation de l'effort.

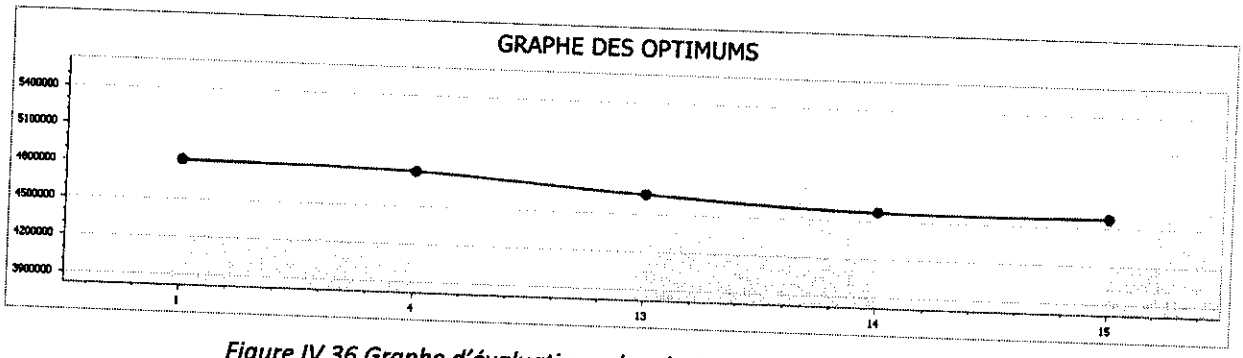


Figure IV.36 Graphe d'évaluations des claviers optimales successif.

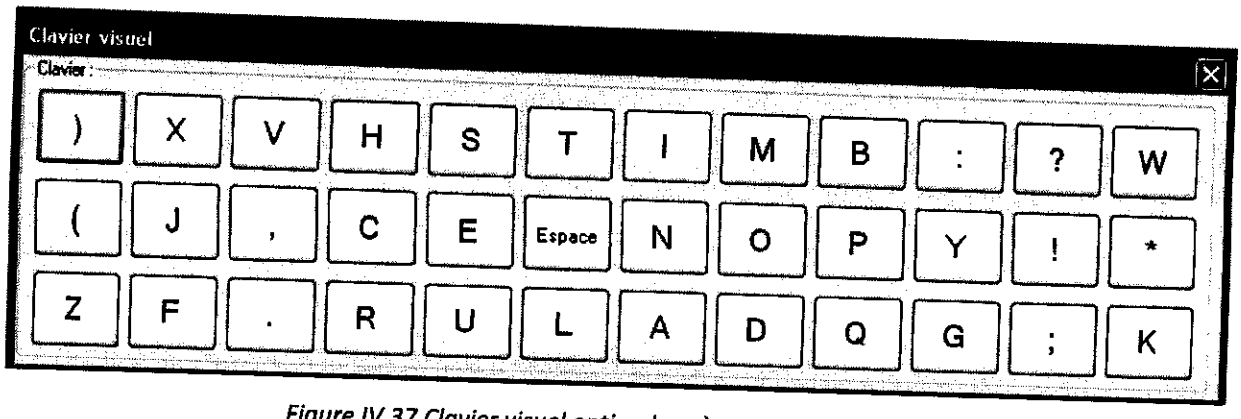


Figure IV.37 Clavier visuel optimal après convergence du système

- P=0,5 (α =5 β=1)

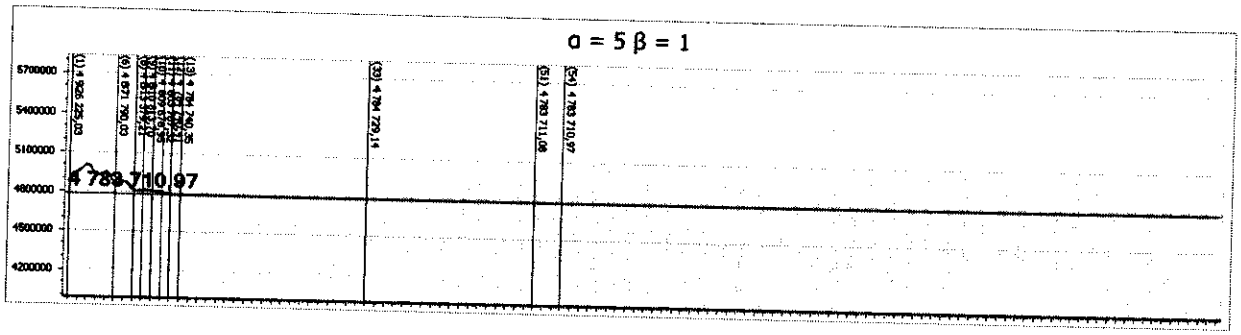


Figure IV.38 Graphe d'évaluation de l'effort.

Convergence prématuré à cause d'une trop grande évaporation.

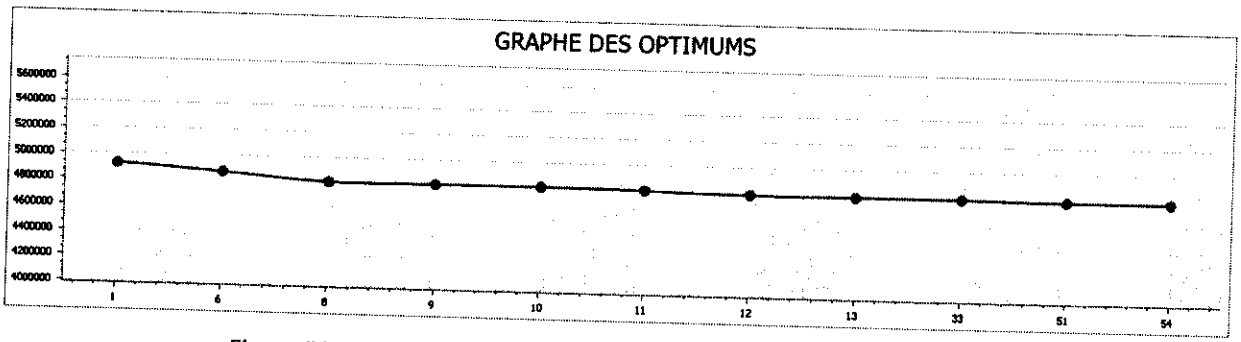


Figure IV.39 Graphe d'évaluations des claviers optimales successif.

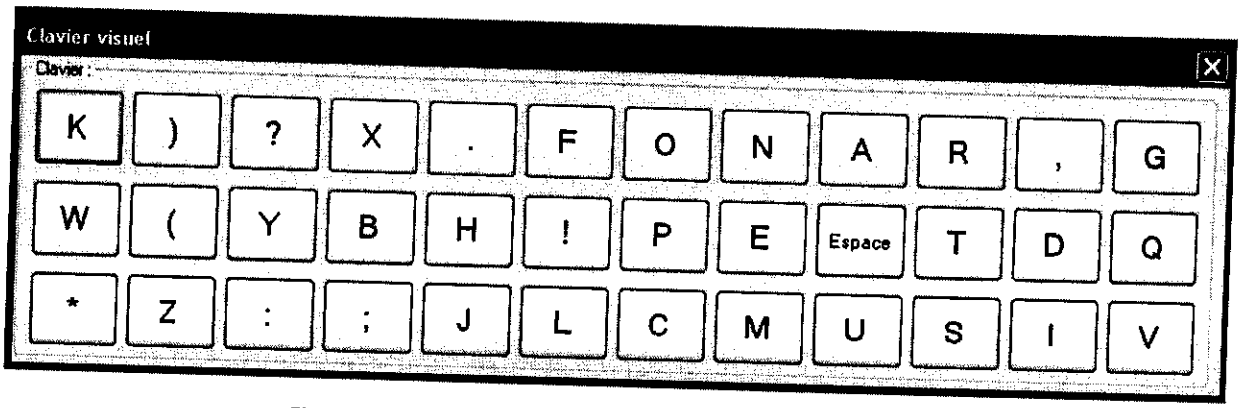


Figure IV.40 Clavier visuel optimal après convergence du système

5eme test:

$\alpha < \beta$

$\alpha = 1$ $\beta = 2$

- p=0,99

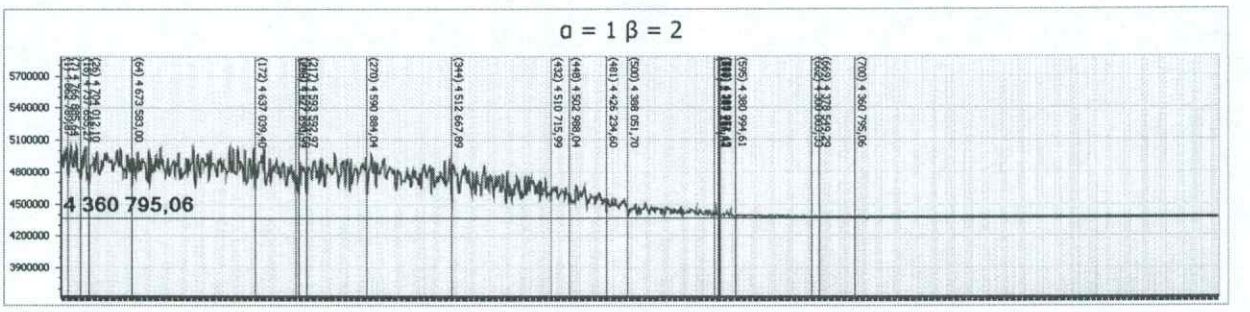


Figure IV.41 Graphe d'évaluation de l'effort.

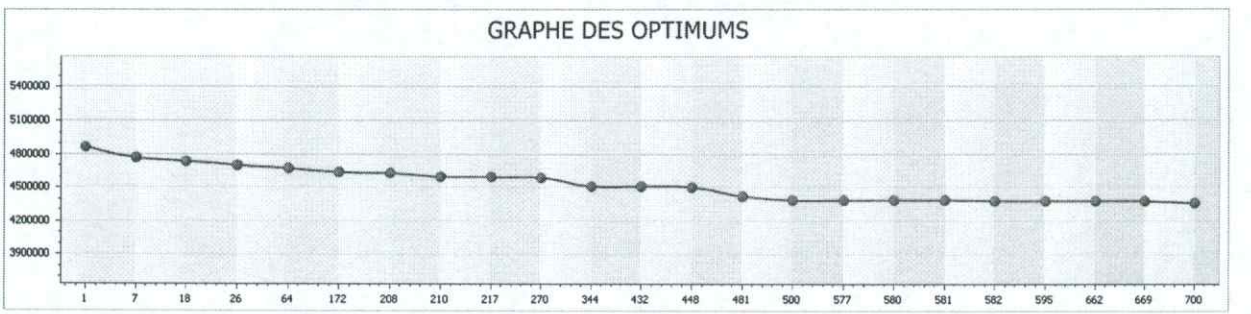


Figure IV.42 Graphe d'évaluations des claviers optimales successif

Figure IV.43 Clavier visuel optimal après convergence du système

- P=0,7 (α =1 β=2)

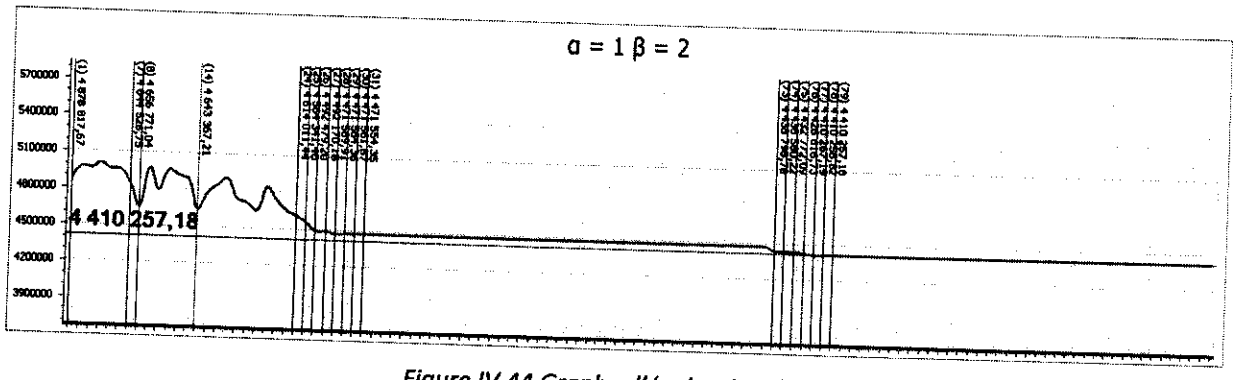


Figure IV.44 Graphe d'évaluation de l'effort.

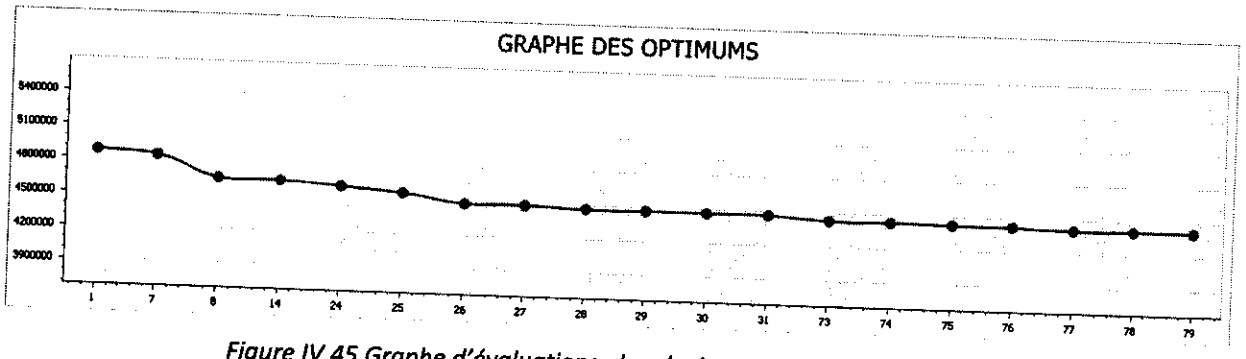


Figure IV.45 Graphe d'évaluations des claviers optimales successif.

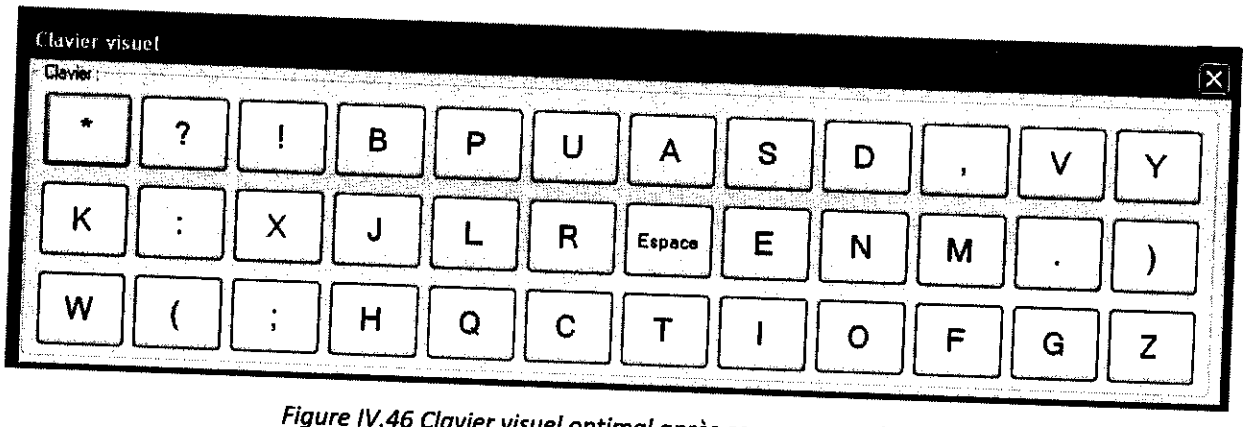


Figure IV.46 Clavier visuel optimal après convergence du système.

- P=0,5 (α =1 β=2)

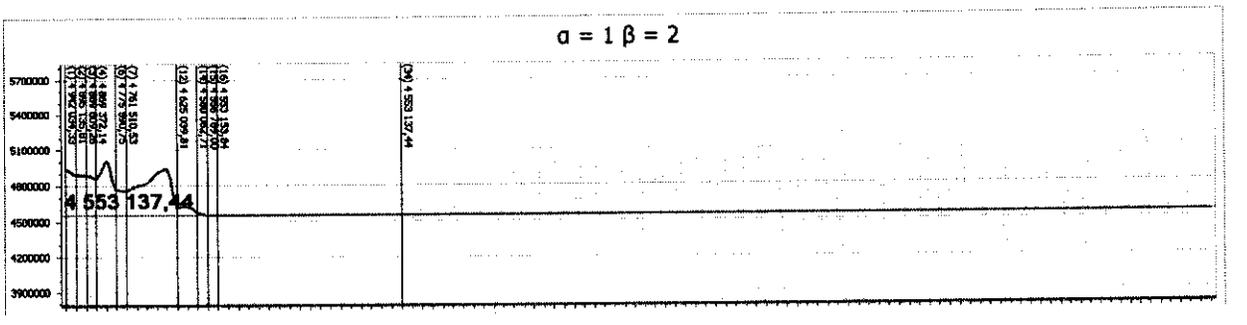


Figure IV.47 Graphe d'évaluation de l'effort.

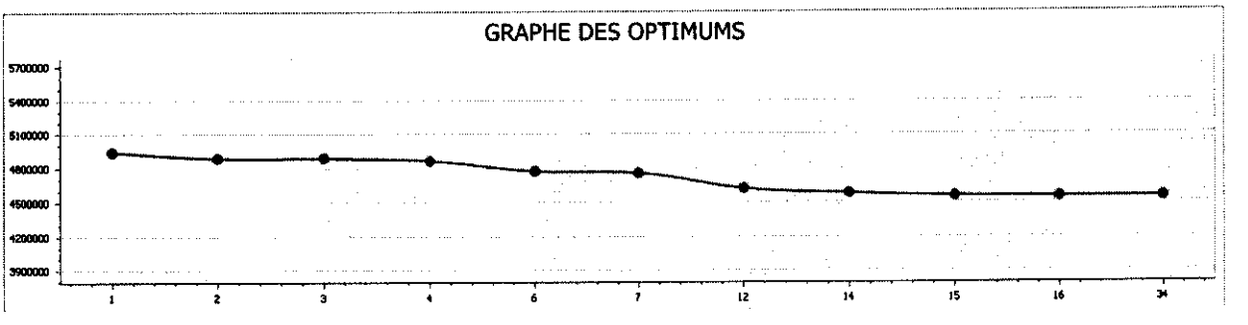


Figure IV.48 Graphe d'évaluations des claviers optimales successif.

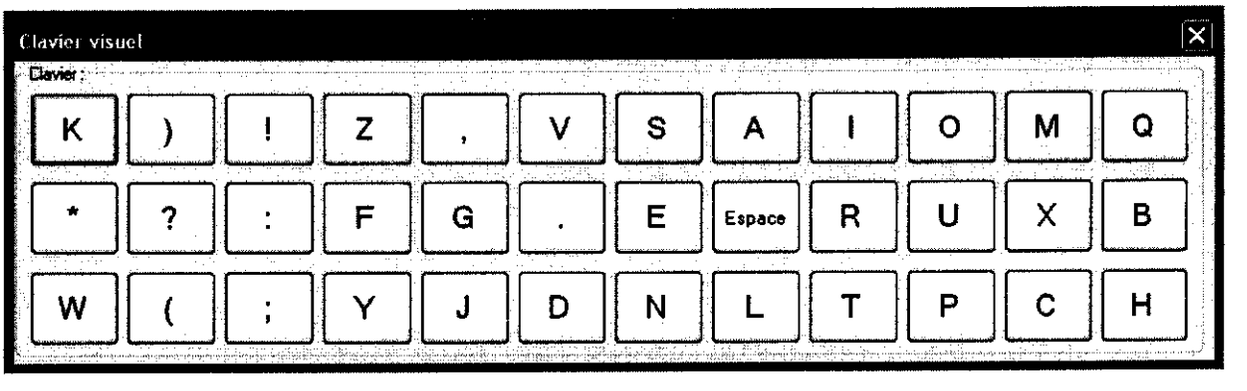


Figure IV.49 Clavier visuel optimal après convergence du système

6ème test :

α =0 β=1

- P=0,99

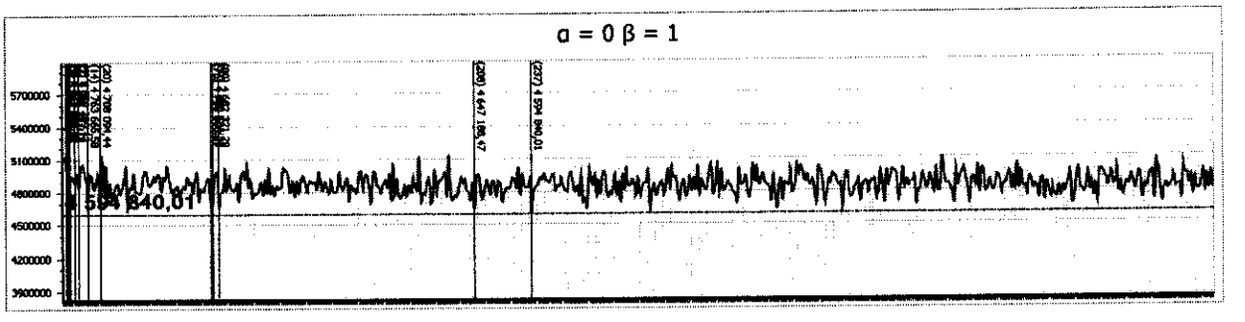


Figure IV.50 Graphe d'évaluation de l'effort.

En supprimant l'attractivité, les fourmis ne prennent en considération que les distances pour calculer les affectations des symboles sur les touches, de ce fait, elle n'on qu'une vision locale du chemin à prendre à chaque fois, ce qui en résulte une grande fluctuation et une absence de convergence.

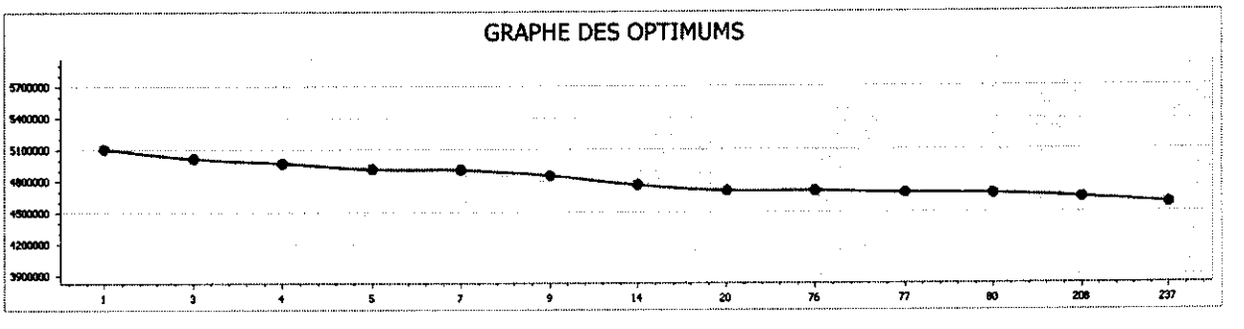


Figure IV.51 Graphe d'évaluations des claviers optimales successif

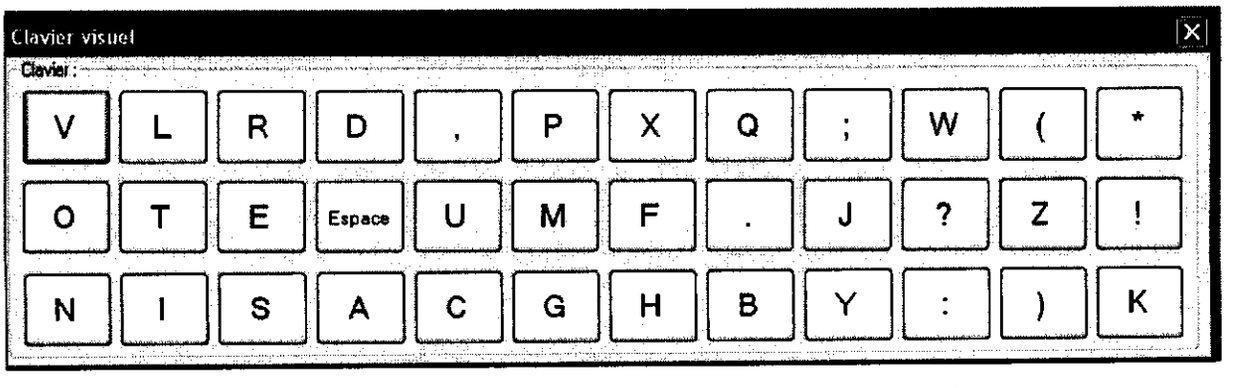


Figure IV.52 Clavier visuel optimal après convergence du système

7ème test :

α =1 β=5

- p=0,99

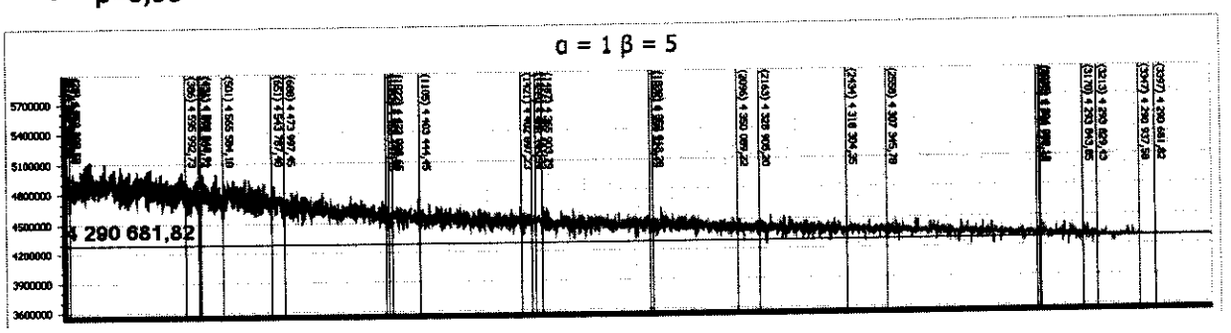


Figure IV.53 Graphe d'évaluation de l'effort.

On remarque une fluctuation durant les premières itérations et un début de convergence au alentour de l'itération 400, pour aboutir à une convergence totale et un résultat final au alentour de l'itération 3300 avec un clavier ayant une évaluation de 4 290 681,82.

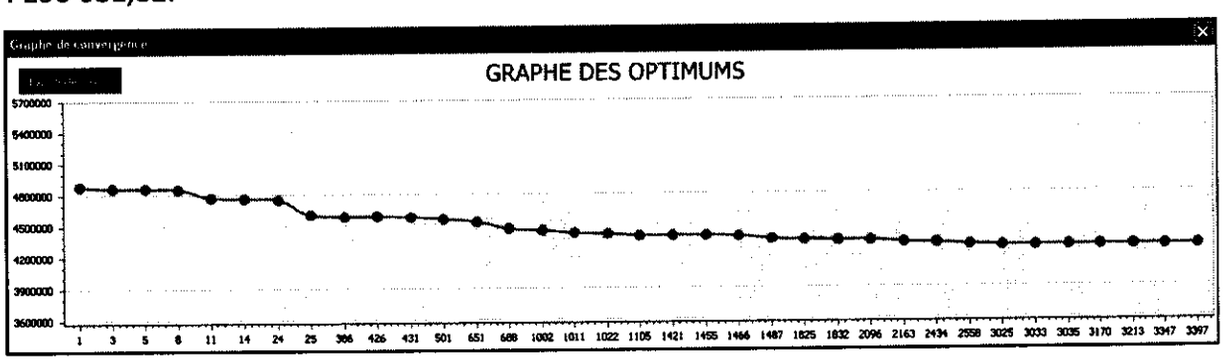


Figure IV.54 Graphe d'évaluations des claviers optimales successif.

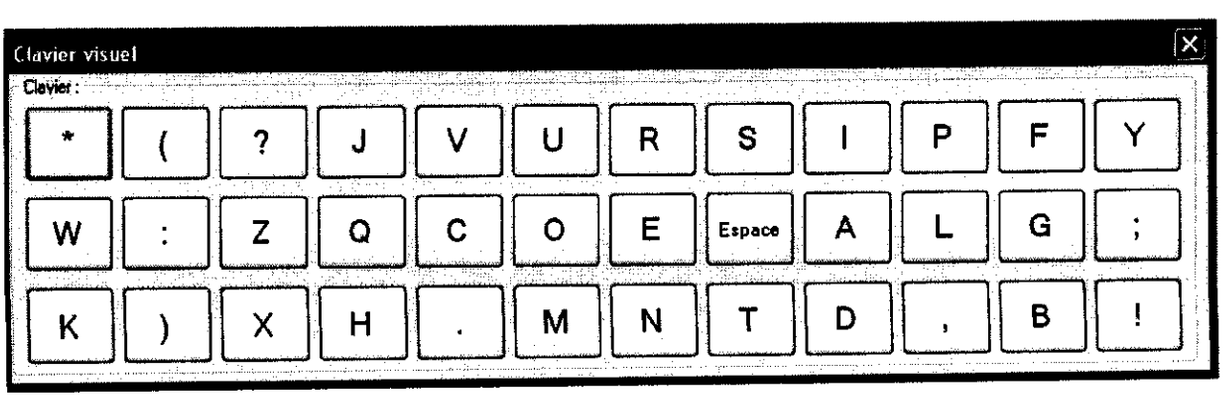


Figure IV.55 Clavier visuel optimal après convergence du système.

- P=0,7 (α =1 β=5)

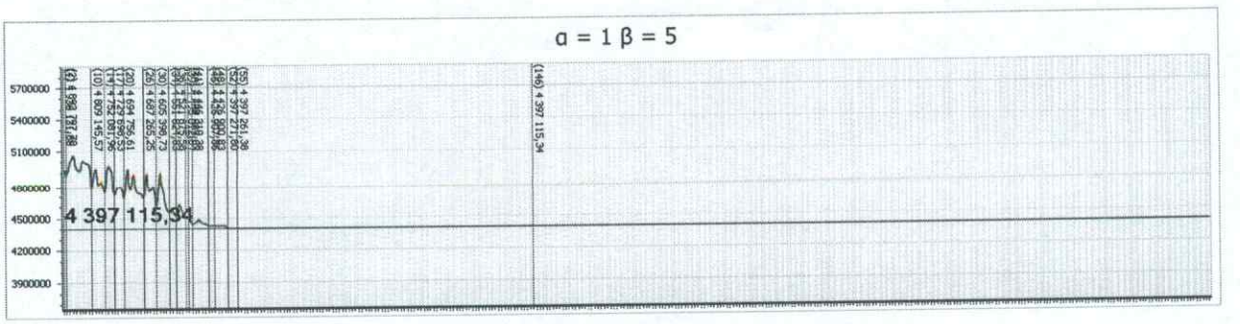


Figure IV.56 Graphe d'évaluation de l'effort.

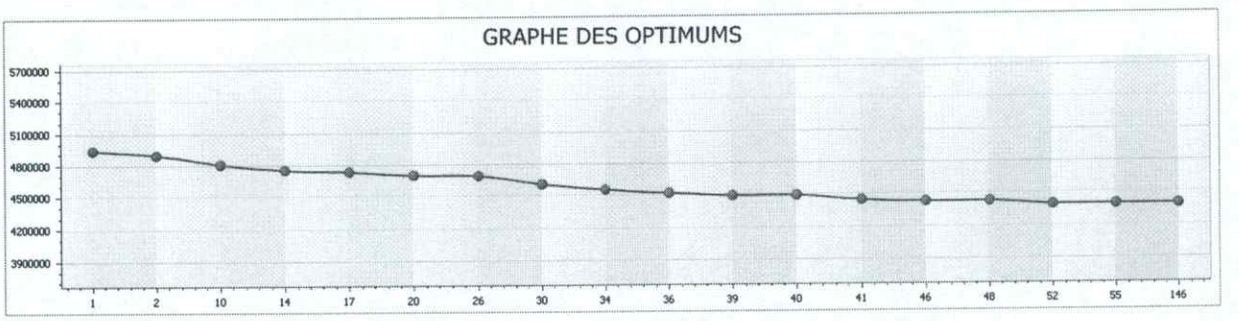


Figure IV.57 Graphe d'évaluations des claviers optimales successif.

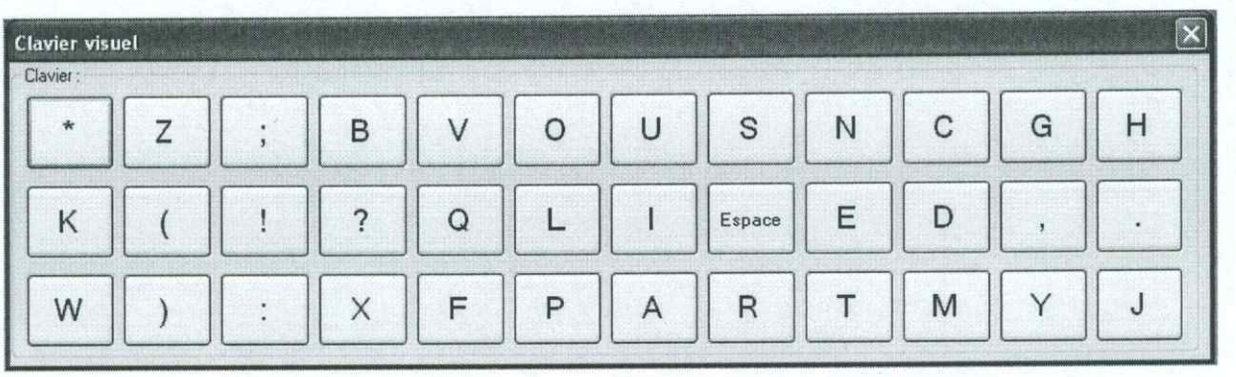


Figure IV.58 Clavier visuel optimal après convergence du système.

- P=0,5 (α =1 β=5)

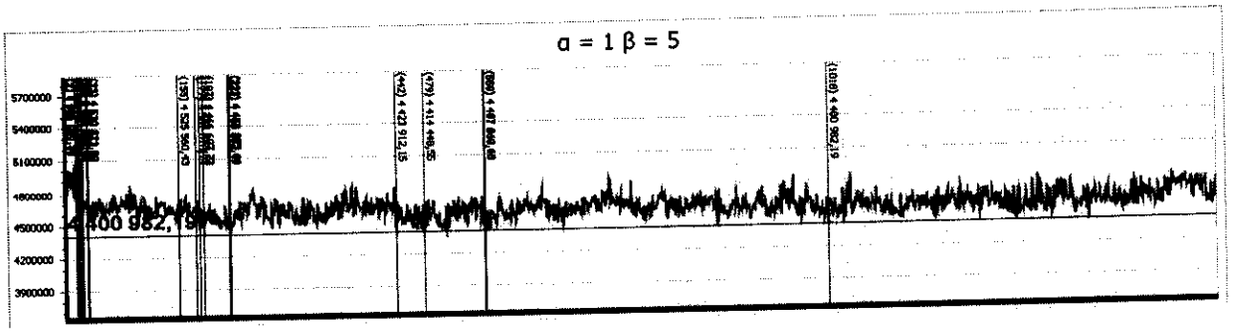


Figure IV.59 Graphe d'évaluation de l'effort.

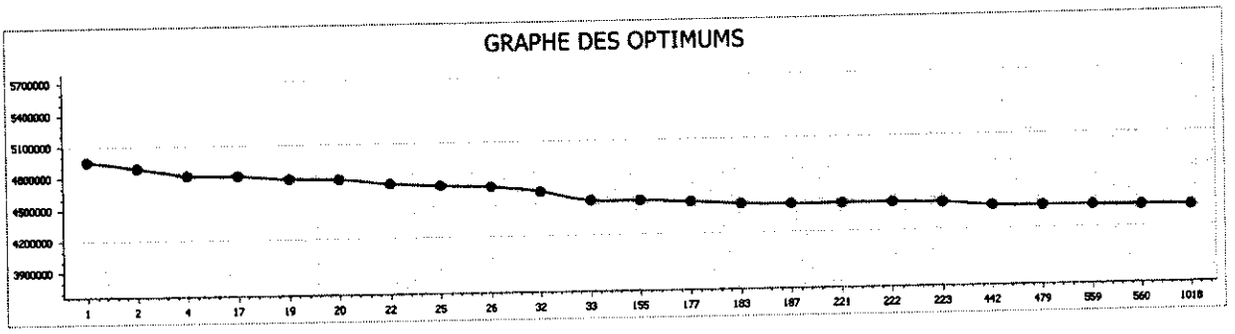


Figure IV.60 Graphe d'évaluations des claviers optimales successif.

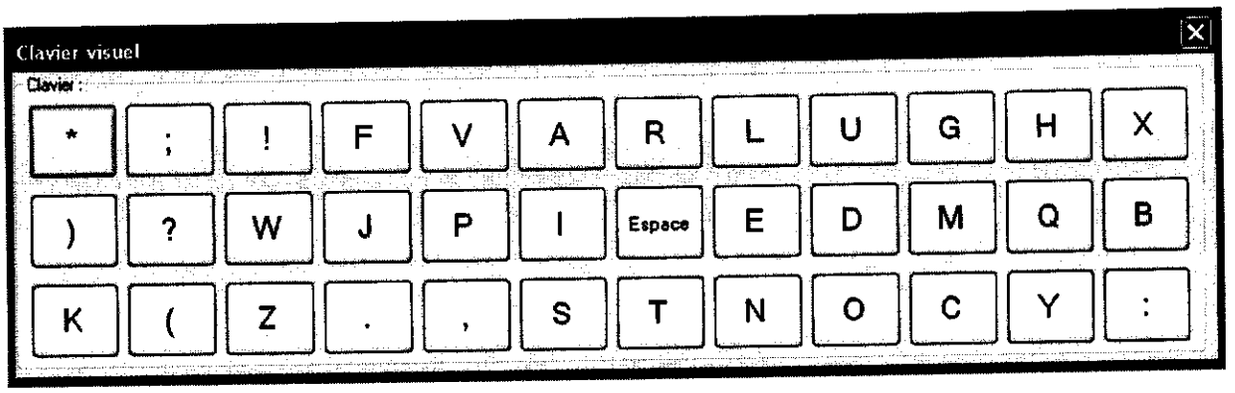


Figure IV.61 Clavier visuel optimal après convergence du système

A	B	P	VALEUR
1	1	0,5	4 592 311,81
		0,7	4 578 735,57
		0,99	4 373 970,02
2	1	0,5	4 684 246,41
		0,7	4 564 552,34
		0,99	4 397 274,83
1	0	0,99	6 874 055,14
5	1	0,5	4 783 710,97
		0,7	4 575 643,85
		0,99	4 428 996,34
1	2	0,5	4 553 137,44
		0,7	4 410 257,18
		0,99	4 360 795,06
0	1	0,99	4 594 840,61
1	5	0,5	4 400 982,19
		0,7	4 397 115,34
		0,99	4 290 681,82

Figure IV.62 Tableau récapitulatif des résultats obtenus lors de la phase de tests.

IV.3. Interprétation des résultants

Durant les tests effectués, nous avons remarqué qu'un compromis entre l'intensification (Contrôlé par le paramètre α) et l'exploration (contrôlé par le paramètre β) est nécessaire, on remarque aussi que l'élimination de l'une d'elle ($\alpha = 0$, ou $\beta = 0$) ne donne pas de résultats significatifs *test 3 et test 6.*

Les meilleurs résultats obtenues ont eu lieu avec une évaporation de phéromones faible ($P = 0,99$).

L'utilisation d'une évaporation trop grande a donné lieu à une convergence prématurée sur les différant tests qu'on a effectué (p=0,7 et p=0,5).

Et d'une façon générale, on remarque que les meilleurs résultats ont été obtenues lors des convergences lentes et cela a eu lieu lorsque on a privilégié l'exploration à l'exploitation $\alpha < \beta$.

IV.4. Comparaison avec le clavier AZERTY

La disposition AZERTY tient son nom de la position des six premières lettres de la première rangée des touches alphabétiques, cette configuration est la plus courante sur les clavier à vocation francophone, et elle à été mise en place à la fin du XIX^{e} siècle sur les machines à écrire pour minimiser les blocages des touches de ces dernières.

L'évaluation du clavier AZERTY sur notre texte d'expérimentation est de : 7 786 372,81

Alors que l'évaluation du meilleur clavier que nous avons obtenu est de: 4 268 133,26

IV.5. Optimisation du clavier avec diverse types de texte

IV.5.1. Texte informatique (texte de programmation en C #)

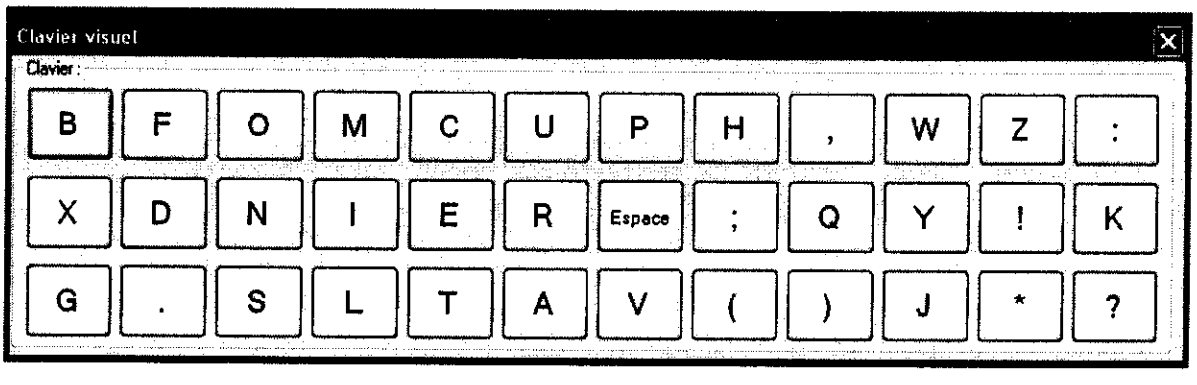


Figure IV.63 Clavier visuel obtenu avec un texte de programmation.

On remarque dans la figure que les symboles « espace », « (», «) », « ; », « , » ont pris tous une position centrale par rapport au reste, cela est dû à leur fréquente utilisation dans les textes de programmation en langage *C#.*

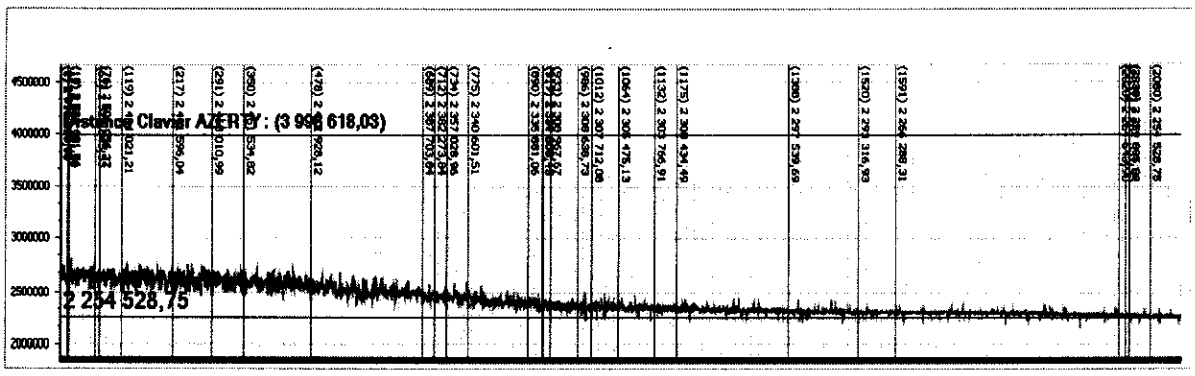


Figure IV.64 Graphe d'évaluation de l'effort sur un texte de programmation.

L'évaluation du clavier AZERTY sur notre texte est de : 3 998 618,03

Et l'évaluation du meilleur clavier obtenu est de : 2 254 528,75

IV.5.2. Texte littéraire de langue anglaise (Hamlet William Shakespeare)

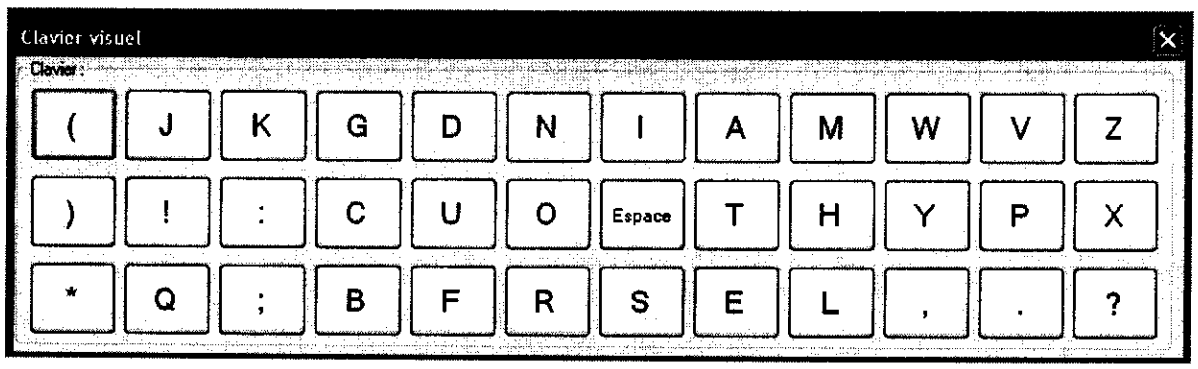


Figure IV.65 Clavier visuel obtenu avec un texte littéraire « Hamlet William Shakespeare »

Ce clavier a été obtenu en utilisant un texte de 27 000 caractères.

On remarque que le symbole « Espace » qui est le plus courant dans ce texte a pris une position centrale proche de toutes les autres touches. Aussi que les touche « T », « H » qui est une combinaison très courante dans la langue anglaise ont pris des positions très proche l'une de l'autre, c'est le cas aussi du couple (W , H).

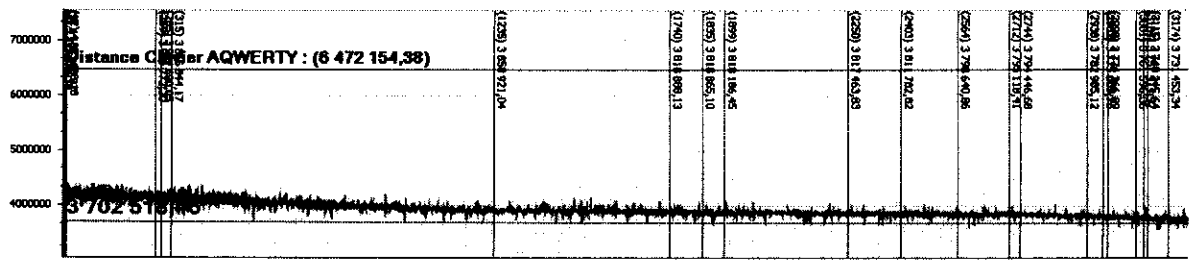


Figure IV.66 Graphe d'évaluation de l'effort sur un texte littéraire anglais.

L'évaluation du clavier QWERTY sur notre texte est de : 6 472 154,38

Alors que l'évaluation du meilleur clavier que nous avons obtenu est de : 3 702 518, 46

IV.6. Conclusion

A travers les tests effectués, nous avons essayé de montrer l'influence de chaque paramètre sur le résultat final de notre algorithme. Les résultats de ces tests nous ont permis d'obtenir à chaque fois un clavier dont l'optimalité varie d'une configuration à une autre.

Nous avons remarqué que, sur l'ensemble des tests, une convergence a eu lieu la plus part des fois, cela est dû, au fait, que nous avons utilisé la variante élitiste de l'AS, qui par définition favorise l'exploitation à l'exploration.

Notons que les travaux présentés ci-dessus sont les plus significatifs et ne sont qu'un petit échantillon des tests qu'on a effectués.

Conclusion générale

Nous avons réalisé, à travers ce document, un algorithme basé sur AS pour résoudre un problème d'optimisation combinatoire : la disposition des touches d'un clavier visuel.

Dans ce travail, nous avons présenté pour commencer : l'optimisation, plus précisément : l'optimisation combinatoire, tout en mettant la lumière sur les méthodes de résolution tel que les méthodes exactes et les métaheuristiques.

Nous avons présenté aussi, des travaux s'inspirant du comportement des fourmis réelles, Les fourmis possèdent en effet de nombreuses caractéristiques collectives et individuelles pouvant nous aider à la résolution de problèmes complexes.

Les premiers travaux apparus dans ce domaine se sont inspirés de l'intelligence collective des fourmis pour la résolution de l'un des problèmes combinatoires d'optimisation : la recherche du plus court chemin. Il est intéressant de remarquer que bien qu'une seule fourmi soit capable de construire une solution (i.e. de trouver un chemin du nid à la nourriture), c'est seulement le comportement de l'ensemble de la colonie qui crée le chemin le plus court. Dans ce contexte nous avons présenté plusieurs modèles d'algorithme AS parmi lesquels nous avons choisi « AS et élitisme ».

« AS et élitisme » favorise l'exploitation à l'exploration, nos tests nous ont montré que dans la majorité des cas le système convergeait, nous avons constaté que les meilleurs résultats que nous avons eu ont été obtenus lors des testes à la convergence lente.

Toutefois nous déplorant le fait de ne pas pouvoir comparer nos résultats avec d'autres résultats, à cause de l'inexistence d'étude détaillé sur ce sujet.

Liste des figures

Références bibliographiques

[AARTS & LENSTRA, 1997] E.H.L. AARTS, J.K. LENSTRA (Eds.), Local search in combinatorial optimization, John Wiley & Sons, 1997.

[algo wiki, 2008] Algorithmes génétiques, Wikipedia, encyclopédie, 20 septembre 2008.

[Berthiau & Siarry, 2001] G.Berthiau, P.Siarry, « Etat de l'art des méthodes d'optimisation *globale* »2001.

[Brossut, 1996] Brossut, R. (1996). *Phéromones, la communication chimique chez les animaux*. CNRS editions, Belin.

[Bullnheimer & Hartl & Strauss, 1999] B. Bullnheimer, R.F. Hartl, and C. Strauss, A new rank-based version of the ant system: a computational study, Central European Journal of Operations Research 7 (1) (1999), 25-38.

[Bouri & Zeblah & Ghoraf & Hadjeri & Hamdaoui, 2005] S.Bouri., A. Zeblah, A. Ghoraf, S. Hadjeri, H. Hamdaoui, Ant Colony Optimization to Shunt Capacitor Allocation in Radial Distribution Systems Acta and Electronica et informatic journal Schekoslovacia N°4,Vo5; 2005.

[Bonabeau & Dorigo & Theraulaz, 2000] E. Bonabeau, M. Dorigo, G. Theraulaz, Nature, Volume 406, Number 6791, Pag. 39 -42 (2000).

[Bianchi & Gambardella & Dorigo, 2002] L. Bianchi, L.M. Gambardella, M.Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. In Proceedings of PPSN-VII, Seventh Inter17 national Conference on Parallel Problem Solving from Nature Science. Springer Verlag, Berlin, Germany, 2002.

[Bentley, 1992] J.L. Bentley, Fast algorithms for geometric traveling salesman problem, ORSA Journal on Computing, vol. 4, pp. 387-411, 1992.

[Bullnheimer & Hartl & Strauss] B. Bullnheimer, R.F. Hartl, and C. Strauss, Applying the ant system to the vehicle routing problem, In: Voss.

[Blum & Sampels, 2002] C. Blum, M. Sampels, Ant colony optimization for FOP shop scheduling: a case study on different pheromone representations, in Proceedings of the 2002 congress on Evolutionary Computation, Honolulu, USA.

[Corne & Dorigo & Glover, 1999] D. Corne, M. Dorigo and F. Glover, editors, New Ideas in Optimization, McGraw-Hill, 11-32.

[Collette and Siarry, 2002] Y. Colette et P. Siarry « Optimisation multiobjectif ». Eyrolles,Paris 2002.

[Chen & Smith, 1996] S. Chen, S. Smith., Commonality and genetic algorithms. Technical Report CMU-RITR- 96-27, The Robotic Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 1996.

[Colorni & Dorigo & Maniezzo, 1991] A. Colorni, M. Dorigo, and V. Maniezzo, Distributed optimization by ant colonies, Proceedings of ECAL'91, European Conference on Artificial Life, Elsevier Publishing, Amsterdam, 1991.

[Cordon & Fernandez de Viana & Herrera & Moreno, 2000] O. Cordon, I. Fernandez de Viana, F. Herrera, L. Moreno, A new ACO model integrating evolutionary computation concepts: the best-worst ant system, in Proceedings of ANTS2000 - from ant colonies to artificial ants, Université Libre de Bruxelles.

[Dréo, 2003] J.Dréo : Métaheuristique pour l'optimisation difficile 2003.

[di Caro & Dorigo, 1998] G. di Caro and M. Dorigo, AntNet: distributed stigmergetic control for communications network , Journal of Artificial Intelligence Research (JAIR), Vol. 9, Pag. 317- 365, 1998.

[Dorigo & Di Caro, 1999] M. Dorigo and G. Di Caro (1999). The Ant Colony Optimization Meta-Heuristic. In.

[Dorigo & Di Caro & Gambardella, 1999] M. Dorigo, G. Di Caro & L.M. Gambardella (1999). Ant Algorithms for Discrete Optimization. Artificial Life, 5(2):137-172.

[Dorigo & Gambardella, 1997] M. Dorigo and L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transaction on Evolutionary Computation 1(1997), 53--66.

[Dorigo & Colorni & Maniezzo, 1996] M. Dorigo, V. Maniezzo, and A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics-Part B 26(1) (1996), 29--41.

[Dorigo and Stützle, 2003] Dorigo, M. and Stützle, T. (2003). Handbook of Metaheuristics, volume 57 of International series in operations research and management science,- 148 -chapter The Ant Colony Optimization Metaheuristics :

[Monmarché, 2000] N. Monmarché, Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation, thèse de doctorat, l'Université de Tours, le 20 décembre 2000.

[Meziane & Hamdaoui & Rahli & Zeblah, 2003] R. Meziane, H. Hamdaoui, M. Rahli, and A. Zeblah Structure Optimization of Electrical Power Network Using Ant Colony Approach. Facta Univ. Ser.: Elec. Energ., vol. 16, No. 2, August 2003, pp. 233-250.

[Meta Wiki, 2008] Méta heuristique, Wikipédia encyclopédie, 20 septembre 2008.

[Ouiddir & Rahli & Meziane & Zeblah, 2004] R. Ouiddir, M. Rahli, R. Meziane and A. Zeblah. Ant colony Optimization for New Redesign Problem of Multi-States Electrical Power Systems. Journal Of Electrical Engineering, Vol 55, N° 1-2, 2004, pp 1-7, ISSN 13-35-36-32 FEISTU.

[Roux, 2001] O. Roux, La mémoire dans les algorithmes à colonie de fourmis : applications à l'optimisation et à la programmation automatique, thèse de doctorat de l'Université du Littoral Cote d'Opale, 2001.

[Stützle and Hoos, 1997] Stützle, T. and Hoos, H. (1997). Improvements on the Ant System: Introducing MAX-MIN Ant System. In Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms, Vienna. Springer-Verlag.

[Stützle and Hoos, 2000] Stützle, T. and Hoos, H. (2000). MAX-MIN Ant System. Future Generation Computer System, 16 :889-914.

[Taillard, 1998] Taillard, E. D. (1998). Programmation à mémoire adaptative et algorithmes pseudo-gloutons : nouvelles perspectives pour les méta-heuristiques. Thèse d'habilitation à diriger les recherches, Université de Versailles Saint Quentin en Yvelines,France.

[Taillard et al., 1998] Taillard, E. D., Gambardella, L. M., Gendreau, M., and Potvin, J.-Y. (1998). Adaptive Memory Programming: A Unified View of Meta-Heuristics. European Journal of Operational Research, 135(1) :1-16.

[Talbi, 2004] TALBI EL GHAZALI « Métaheuristiques pour l'Optimisation Combinatoire Multi-objectifs », Laboratoire d'Informatique. Fondamentale de Lille,Université de Lille 1, France.

[Vander Meer & Breed, 1998] R. Vander Meer, M. Breed, K.E., E., and M.L., W., editors (1998). *Pheromone Communication in Social Insects*. Westview Press.

[Larousse, 2002]