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

Université Saâd DAHLAB de Blida

Faculté des Sciences

Département d'Informatique

Mémoire présenté par :

M^lle BACHIR *Chahrazed*
M^lle HERIZI *Imen*

En vue d'obtenir le diplôme de Master

Domaine : Informatique

Filière : Informatique

Spécialité : Génie logiciel

Sujet :

Coopération des méthodes d'optimisation pour la résolution du problème de repliement de protéines

Soutenue le :	**, devant le jury composé de :**
M.	**Président.**
M.	**Rapporteur.**
M.	**Examinateur.**
M^me S.BENSETTITI	**Promotrice.**

Dédicace

Je dédie cette humble œuvre à mes parents en guise de témoignage de gratitude pour avoir tant peiner a me faciliter les dures taches de l'apprentissage, à mon fiancé Ahmed, à mon cher neveu Zaki, à mon binôme Imen qui à partagé avec moi les moments les plus durs et les plus beaux de ce voyage et Enfin à toute ma famille et à tous mes amis.

Chahra

Chahra

Dédicace

Je dédie cette humble œuvre à mes parents en guise de témoignage de gratitude pour avoir tant peiner a me faciliter les dures taches de s'apprentissage, à mon mari Abdou, à ma chère petite fille Yara , a mon binôme Chahra qui à partagé avec moi les moments les plus durs et les plus beaux de ce voyage , à ma précieuse amie Fatima Enfin à toute ma famille et à tous mes amis.

Imen

Imen

Remerciement

Le devoir de reconnaissance nous appelle à remercier Mme Benstiti notre promotrice pour nous avoir prises en charge.

Mrs messied le chef de département informatique et Mrs Ayet akasha pour le soutien et le courage dont ils nous ont dotées et la compréhension dont ils ont fait preuve.

Comme nous tenons à remercier Mrs Ouldaissa et Mlle Ameur Khadija et Mrs Chennaf Mohamed pour leur aide.

Ainsi que tous nos professeurs pour leur patience et leurs précieux conseils.

Résumé

La majorité des problèmes d'optimisation combinatoire sont NP-complet. Tenter de les résoudre avec une méthode exacte s'est avéré irréalisable du fait du temps de calcul excessif.

Des méthodes approchées ont été développées pour une résolution rapide de ces problèmes, le temps de calcul que mettent les méthodes approchées est considéré comme raisonnable par rapport au temps de calcul excessifs des méthodes exactes.

Parmi les méthodes approchées, Les métaheuristiques.

Les métaheuristiques sont efficaces pour la résolution des problèmes d'optimisation, elles offrent un moyen efficace pour atteindre une grande performance de calcul.

Dans notre travail, on voudrait montrer que la collaboration/hybridation de méthodes permet d'améliorer significativement la qualité des solutions obtenues et de traiter des problèmes de grande taille, nous nous intéressons à la résolution approchée du problème de repliement des protéines, qui est un problème largement étudié dans la littérature.

Nous avons proposé une adaptation d'une nouvelle méta-heuristique colonies de fourmis hybridé avec une recherche locale, que nous l'avons appliqué à notre problème.

Mots Clés : *Optimisation combinatoire, métaheuristiques, repliement de protéines, Colonies de fourmis (ACO), recherche locale, Modèle HP.*

TABLE DES MATIERES

Liste des figures

Liste des tableaux

INTRODUCTION GÉNERALE

Introduction générale

Il est généralement admis que les différentes méthodes d'optimisation possèdent chacune des avantages et des inconvénients, des forces et des faiblesses.

Pour rendre les algorithmes plus performants et fiables, de nombreux travaux proposent de combiner différentes heuristiques et/ou méthodes exactes.

On voudrait montrer que la collaboration/hybridation de méthodes permet d'améliorer significativement la qualité des solutions obtenues et de traiter des problèmes de grande taille. Nous avons abordé cette question pour un problème bioinformatique bien connu, qui est celui du *repliement de protéines*.

Le repliement des protéines est un des problèmes majeures de la biologie moléculaire, chaque protéine est un petit polymère qui est capable d'exister sous deux formes, l'une déplier (dénaturé) et biologiquement inactive, l'autre replier (native) est biologiquement active, Déterminer la conformation native de la protéine dans cet état de moindre énergie est connu sous l'appellation de *problème du repliement de protéines*. Celui-ci peut être modélisé comme un problème d'optimisation combinatoire. Son étude et sa résolution revêtent un intérêt dans différents domaines, comme la santé humaine et la biotechnologie. Dans le domaine de la santé humaine, savoir résoudre ce problème de manière exacte pourrait contribuer à la prédiction et au traitement des maladies comme la maladie d'Alzheimer. En biotechnologie, il pourrait permettre par exemple la réalisation de procédés enzymatiques à grande échelle, le développement de médicaments protéiques (Insuline, HGH).

Notre projet consiste donc, à modéliser le repliement d'une protéine en un problème d'optimisation combinatoire et appliquer par la suite des méthodes de résolution approchée de ce problème. Plus particulièrement nous appliquerons un algorithme hybride entre l'algorithme de colonies de fourmis et un algorithme de recherche locale.

Ce mémoire est subdivisé en quatre chapitres :

Le premier chapitre fournit en détail certaines notions sur les protéines et leur structure ainsi que la présentation de problème du repliement de protéine et les différents modèles existants.

Le second chapitre évoque des notions essentielles d'optimisation combinatoire et de la théorie de la complexité. On présente par la suite les différentes méthodes de résolutions proposées pour la résolution de tels problèmes. Il décrit une présentation détaillée des métaheuristiques des colonies de Fourmies et la recherche locale, Il expose les combinaisons de coopération/hybridation entre ces deux métaheuristiques.

Le troisième chapitre consiste à la modélisation et l'application des méthodes de résolution sur le problème de repliement de protéine.

Le quatrième chapitre sera réservé à l'implémentation et au résultat obtenu ainsi qu'une petite présentation sur le logiciel utilisé.

Enfin On achèvera par une conclusion et les perspectives d'un tel travail.

CHAPITRE1

LE REPLIEMENT DE PROTÉINE

Problématique et modélisation

1. Introduction

Le problème du repliement de protéine est un problème ouvert dans le domaine de chimie et science de protéine. Il attire de plus en plus l'attention des chercheures en bioinformatique.

Une fonction de protéine est étroitement rapprochée de sa structure tridimensionnelle et donc pour déterminer comment une protéine fonctionne il faut connaitre sa conformation tridimensionnelle. Le problème du repliement consiste à prédire cette dernière a partir de la séquence d'acides aminés qui forment la protéine.

Ce chapitre définit la structure d'une protéine, il donne une présentation du problème de repliement de protéine en s'appuyant sur le modèle HP.

2. Les protéines

2.1 Définition

Une protéine, aussi appelée ***protéide***, est une macromolécule biologique composée par une ou plusieurs chaine(s)(ou séquence(s)) d'acides aminés liés entre eux par des liaisons peptidique. En général, on parle de protéine lorsque la chaîne contient plus de 50 acides aminés. Dans le cas contraire, on parle de peptides et de polypeptides, mais plus souvent simplement de (petite protéine).

Un être vivant fabriquerait au total plus de 100 000 sortes différentes de protéines. Chaque cellule fabrique en moyenne de 15 000 sortes différentes. Prés de 50% du poids sec d'un être vivant est fait de protéines.

2.2 Fonction des protéines

Les protéines sont des macromolécules biologiques qui interviennent dans la grande majorité des processus qui régissent le fonctionnement de tout être vivant. Les rôles joués par les protéines au sein d'un organisme sont aussi variés que complexes.

Certaines protéines, appelées enzymes, agissent en tant que catalyseurs et augmentent avec une spécificité remarquable, les vitesses des multiples réactions indispensables a la survie de l'organisme. Ce sont également des protéines qui servent au stockage et au transport de petites molécules, qui interviennent dans le processus de la photosynthèse, qui contrôlent le passage de molécules à travers des membranes lipidiques qui délimitent Les cellules et leurs compartiments, ou qui, en tant qu'hormones, transmettent l'information et permettent la régulation de processus cellulaires complexes. Diverses protéines - dont notamment les anticorps- sont affectées au système immunitaire et permettent à l'organisme de se défendre contre les intrusions bactériennes ou virales. Les protéines sont également des composantes majeures des systèmes de conversion d'énergie chimique en énergie mécanique, tels que les muscles.

Notons finalement que de nombreuses protéines ont simplement un rôle structurel et fournissent l'architecture filamenteuse indispensable à l'organisation des cellules et à la génération de tissus tels que les os, les cheveux ou les ongles. **[LIN07]**

Les protéines remplissent des fonctions très diverses :

- ***Catalyse*** (ex. : de nombreuses *enzymes*- les principaux catalyseurs biologiques - sont des protéines)
- ***Transport*** (ex. : *l'hémoglobine* transporte l'*oxygène* des *poumons* aux organes).
- ***Communication*** (ex. : de nombreuses *hormones*- comme l'*insuline* - sont des protéines et peuvent transporter un message à travers tout l'organisme)
- ***Signalisation*** (ex. : des protéines sont impliquées dans le *chimiotactisme*)
- ***Reconnaissance*** (ex. : les systèmes *immunitaires* possède des protéines spéciales - *les immunoglobulines*- qui permettent la reconnaissance moléculaire de formes

étrangères c'est-à-dire n'appartenant pas aux formes moléculaires de l'organisme qui les fabrique). **[LIN07]**

2.3 *Structure de protéine* [LIN07]

La séquence chimique d'acides aminés, dont l'élément de base est la chaîne C-C-N est aussi appelée **structure primaire** d'une protéine, elle représente l'état dénaturé de la protéine, dans cette état la protéine est complètement instable.

Certaines parties de la chaîne d'acides aminés adoptent une structure régulière appelée **structure secondaire,** formé à partir de la structure primaire par un repliement, Cette structure est plus stable par rapport à la structure primaire. C'est une structure intermédiaire pour le repliement entre *l'état dénaturé* et *l'état natif* (plus stable).

On reconnaît deux grands types de structure secondaire :

- **L'hélice alpha**

Dans la structure dite en hélice alpha, la chaîne d'acides aminés prend la forme d'un tire-bouchon. Les différentes spires sont stabilisées par des liaisons hydrogène ***(Fig1 .2)***.

- **Le feuillet bêta**

Dans un feuillet bêta, il se forme des liaisons hydrogène entre certains segments de la chaîne disposés parallèlement les uns par rapport aux autres. L'ensemble forme comme une membrane plissée ***(Fig1 .2)***.

Une protéine est donc faite d'hélices alpha et de feuillets bêta reliés par des segments qui n'ont pas de structure secondaire particulière.

La forme finale de la chaîne d'acides aminés, c'est à dire la structure tridimensionnelle finale qu'adopte la chaîne d'acides aminés, constitue la **structure tertiaire** de la protéine.

La structure primaire ne fournit malheureusement aucune information sur la fonction de la protéine dans l'organisme. Il faut pour cela déterminer la conformation de celle-ci, c'est-a-dire sa structure tridimensionnelle.

La majorité des protéines se replient pour adopter cette structure tridimensionnelle, dont la particularité est d'utiliser des interactions entre résidus proches dans l'espace mais distants dans la séquence. Cette structure tridimensionnelle également appelée état natif, est stable grâce aux nombreuses interactions favorables qui s''établissent en son sein. Notons que l'on appelle *interactions natives* une interaction présente dans la structure native.

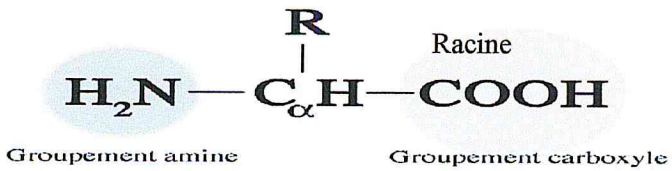


Fig.1 .1 Représentation atomique d'un acide aminé

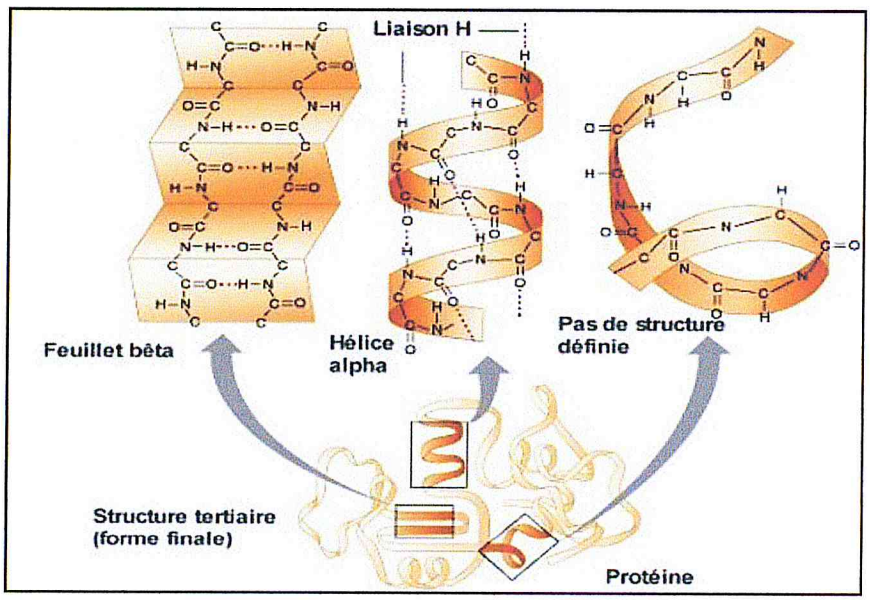


Fig.1 .2 les différents structurent de protéine

3. Le repliement de protéine

3.1 Présentation de problème de repliement de protéine

Une protéine peut adopter différentes conformations, celles de l'*état dénaturé* étant généralement les moins stables. A l'image d'un ressort trop tendu, une protéine en état instable cherche à atteindre son état le plus stable caractérisé par une énergie libre minimale. Une hypothèse largement admise est que *la structure native d'une protéine correspond généralement à sa conformation d'énergie libre minimale.*

L'état natif d'une protéine est atteint dans des conditions typiquement trouvées dans les cellules des êtres vivants (PH quasi-neutre, des températures de 20° à 40°, en solution aqueuse). Les protéines peuvent être dénaturées (autrement dit, perdre leur structure tridimensionnelle native) en les plaçant dans des conditions sortant de ce cadre. Toutefois, ce processus est souvent réversible, et de telles molécules peuvent retrouver leur état natif sous des conditions contrôlées. **[CAO08]**

Le repliement d'une protéine est un processus extrêmement spécifique et particulièrement efficace. En effet, la rapidité avec laquelle une protéine trouve sa conformation de plus basse énergie est remarquable. Par exemple, une protéine de 100 résidus peut adopter 2^{100} ($\approx 10^{30}$) structures tridimensionnelles différentes en supposant que seulement 2 conformations soient accessibles à chaque résidu, ce qui est loin de la réalité. En admettant que le passage d'une conformation à une autre se fait en 10^{-13} secondes (ce qui correspond au temps nécessaire pour une rotation autour d'une liaison peptidique), il faudrait 10^{17} secondes soit environ trois milliard d'années, pour tester toutes les conformations possibles. **[FAB06]**

Très clairement, la totalité des conformations accessibles à une *véritable* protéine, sans aucune simplification, dépasse de loin le degré d'abstraction humain. Pourtant, les protéines arrivent à retrouver leur structure native en un temps allant de quelques millisecondes à quelques secondes ! Levinthal en 1969 **[DIL97]**, fut le premier à relever cette incompatibilité que l'on nomme aujourd'hui *paradoxe de Levinthal*. Il ajoutait que, de toute évidence, les protéines n'explorent pas la totalité de leur espace conformationnel. Actuellement, seuls des modèles théoriques viennent expliquer la rapidité de ce processus. En effet, les mécanismes qui permettent à une

séquence d'acides aminés de se replier en une structure tridimensionnelle unique sont actuellement mal connus. **[FAB06]**

3.2 Modèles de repliement

- ***Le modèle diffusion-collision*** propose un mécanisme de repliement hiérarchique qui permet de réduire l'espace conformationel. La théorie soutient qu'il y aurait, dans un premier temps, un repliement des sous domaines de la protéine, ces sous-domaines étant d'une taille très limitée. Par la suite, des interactions entre sous-domaines se créeraient, formant ainsi des domaines plus importants. Cela durerait jusqu'à la formation complète de la structure native de la protéine.**[ACO08]**

- ***Le modèle de l'effondrement hydrophobe*** il s'appuie sur la tendance des résidus hydrophobes à se regrouper dans les premiers instants du processus affin d'éviter tout contact avec l'eau. Ce regroupement aurait lieu avant la formation de structures secondaires et aurait une influence prépondérante pour la suite du repliement. En effet, le noyau hydrophobe temporairement formé réduirait considérablement l'espace conformationel accessible à la protéine. **[MBO08]**

- ***Le modèle de nucléation-condensation*** met en avant l'importance des interactions tertiaires (c'est-à-dire entre résidus distants dans la séquence). Afin que ces interactions aient lieu, la formation d'un noyau de repliement est requise. Les fragments de protéine non repliés entrant en contact avec le noyau adopteraient ensuite leur structure native. **[REP00]**

Durant les années 90, un nouveau modèle de repliement protéique a été développé .Il à comme caractéristique principale de représenter le *paysage énergétique* d'une protéine. Il s'agit d'une représentation multidimensionnelle de l'énergie libre d'une conformation de la protéine en fonction de sa similarité avec la structure native.

Une interprétation tridimensionnelle du paysage énergétique produit un entonnoir communément appelé *entonnoir de repliement*. L'axe vertical de cet entonnoir

représente l'énergie libre d'une conformation et les axes horizontaux représentent les conformations de la protéine accessibles à un certain niveau d'énergie. A la base de cet entonnoir, on trouve la structure native qui est donc d'une énergie libre minimale. Plus on monte dans l'entonnoir, plus le nombre de conformations possibles à un même niveau d'énergie augmente. Le repliement serait donc un processus progressif de modifications conformationnelles entraînant la création d'interactions natives favorables et débouchant sur la structure native **(fig.1-3).**

Dans une situation idéale, l'entonnoir serait parfait et chaque modification conformationnelle en direction de l'état natif stabiliserait la protéine **(fig.1-3(a)).**Malheureusement, l'expérience montre que des modifications conformationnelles nécessaires à la création d'interactions natives peuvent être défavorables d'un point de vue énergétique ! Cela introduit des variations dans le paysage énergétique et laisse apparaitre un concept bien connu dans le domaine de l'informatique : le problème des minima locaux **(fig.1-3(b)).** Un paysage énergétique peut donc s'avérer plus ou moins complexe, ce qui induit qu'il existe différents chemins de repliement entre une conformation d'état dénaturé et une conformation d'état stable **(fig1-3(c))**.

Il est intéressant de comparer le paradoxe de Levinthal à un paysage énergétique plat **[HS-97].** L'idée à l'époque était qu'il existait une conformation native unique et que toute autre conformation était inintéressante. On se rend alors compte que le problème d'origine est mal posé : en effet, il est mathématiquement improbable de tomber dans le trou de l'entonnoir à partir d'un paysage plat en un temps raisonnable **(fig1. 3(d)).**

Ce modèle de paysage énergétique ne remet pas en question tous les modèles précédents, au contraire. Il permet d'expliquer plusieurs phénomènes observés tels que l'existence de chemins de repliement préférés. Cela appuierait certaines théories comme par exemple celles de la création d'un noyau hydrophobe ou d'intermédiaires de repliement. Concrètement, le paysage énergétique procure un cadre général dans lequel les cas particuliers peuvent être interprétés.

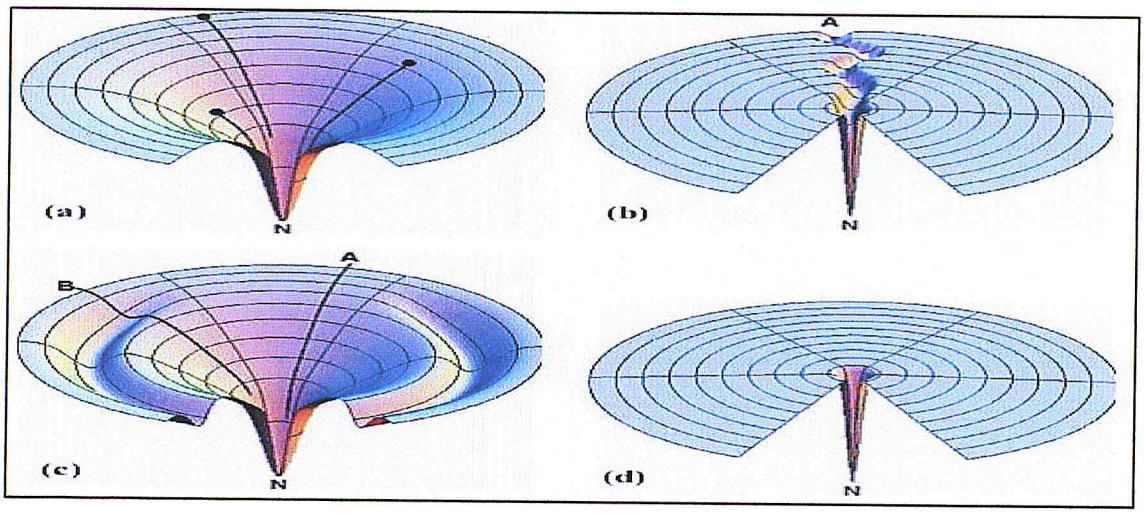


Fig.1.3 Représentation tridimensionnelle du paysage énergétique d'une protéine
(a) *un paysage énergétique idéal* (b) *un paysage énergétique réel*
(c) *deux chemins de repliement différents* (d) *le paradoxe de Levinthal*

4. La prédiction de la structure native des protéines

4.1 Objectif

La structure native d'une protéine est une richesse en soi. Elle est l'inconnue nécessaire à la résolution d'une foule de questions, dont la principal est : « *Comment ça marche* ? ». En effet, pour comprendre le fonctionnement d'une protéine il faut impérativement en connaître la structure native. Par exemple, les sites actifs des protéines ne sont généralement fonctionnels qu'une fois la protéine complètement repliée.

La prédiction de la structure native d'une protéine nous offre plusieurs possibilités :

- l'étude des interactions possibles entre une protéine et diverses molécules permet par exemple la création de médicaments qui agiront précisément là où c'est nécessaire;
- certaines protéines subissent des mutations pour diverses raisons et la compréhension fine de l'effet de ces mutations passe par la connaissance de la structure native ;
- la conception de protéines dont la séquence est modifiée permettrait d'obtenir de nouvelles propriétés comme par exemple une augmentation de la stabilité de la protéine ou encore l'amélioration de réactions enzymatiques dans le milieu industriel ;

- la compréhension des mécanismes de stabilisation de l'état d'une protéine pourrait conduire, dans un futur plus ou moins proche, à la création de nouvelles protéines avec un rôle bien précis et défini à l'avance. Le besoin sera alors à l'origine de la protéine. **[FABO6]**

4.2 Aperçu générale sur les méthodes de prédiction

Il existe deux grandes approches de prédiction de structure :

- Modélisation basée sur des structures connues et une similarité détectable entre la plupart de la séquence à modéliser et la séquence d'une (au moins) structure connue (*Modélisation par Homologie)*

Consiste en 4 étapes :

- Trouver des structures connues qui ont un rapport avec la séquence
- Aligner la séquence à modéliser sur la séquence ou les séquences connues
- Construire le modèle
- Evaluer le modèle

- Les méthodes *Ab initio.* Elles prédisent la structure à partir de la séquence seule, sans recourir à des similarités avec des repliements connus .Basée sur l'idée que la structure native correspond à un minimum globale énergétique. Ces méthodes accomplissent une recherche à grande échelle dans l'espace conformationel. Le but des recherches est de trouver des conformations tertiaires qui on une énergie particulièrement basse pour une séquence donnée.

Les deux caractéristiques les plus importantes de ces méthodes :

- La procédure de recherche doit être rapide et efficace
- La fonction d'énergie libre doit être la plus juste possible. **[FABO6]**

Les impératifs de la perdition des structures natives (ab initio)

Afin de réaliser au mieux une prédiction, certains outils sont nécessaires :

- une modélisation de la protéine étudiée.
- une méthode de discrétisation de l'espace dans lequel elle évolue.

- une méthode de recherche qui permet l'exploration de l'espace conformationel.
- une fonction énergétique permettant l'évaluation d'une structure par rapport à sa séquence.
- dans certains cas, il sera intéressant de posséder d'un moyen d'évaluation des prédictions Obtenues. **[FABO6]**

5. Discrétisation de l'espace conformationel [BMO08]

L'espace conformationel est immensément grand. Diverses techniques de discrétisation existent afin de permettre une diminution substantiellement importante des conformations possibles. Bien entendu, il s'agit à nouveau de trouver un bon compromis entre degré de simplification et qualité de résultat. Deux grands types de discrétisation existent :

- **le *modèle réseau*** propose une restriction de l'espace conformationel en maille fixe. ce modèle peut être représenté par une grille bidimensionnelle ou tridimensionnelle fixe.

Le caractère statique de ces modèles représente un facteur limitatif important au niveau de la qualité des solutions obtenues.

En effet, dans un modèle réseau, on travaille avec des coordonnées fixes définies par la maille utilisée. Ainsi, dans le cas d'une représentation atomique de la protéine, les distances interatomiques seront fixes et les possibilités de rotation angulaire seront également définies (par exemple, dans un modèle bidimensionnel ou tridimensionnelle carré où la maille représente la distance interatomique, ces rotations seront toujours multiples de 90°).

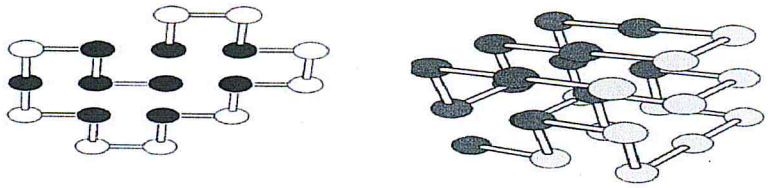


Fig.1.4 Modèle réseau bidimensionnelle gauche de la figure
Modèle réseau tridimensionnelle droite de la figure **[DIL95]**

- Le ***modèle hors réseau*** propose une restriction de l'espace conformationel en limitant par exemple les rotations autour des liaisons chimiques à certaines valeurs angulaire bien définies, ce qui a pour conséquence de créer un modèle réseau dynamique.

6. Modélisation de la protéine [AG08]

Une difficulté récurrente en informatique est celle de la modélisation des problèmes étudiés. Dans ce cas-ci comme dans beaucoup d'autres, il faut trouver un juste compromis entre simplification du problème et qualité des résultats.

La modélisation d'une protéine peut se faire par une représentation complète au niveau atomique des acides aminés la composant. Evidemment, un tel modèle entraînerait un coût de calcul non négligeable. D'un autre côté, une représentation trop simpliste de la protéine entraînera une dégradation de la qualité des résultats. Il faut donc trouver un compromis dont on pourra tirer le meilleur parti dans les deux cas.

6.1 *Le modèle (φ,ψ,ω)*

Les résidus d'une protéine peuvent adopter diverses conformations en fonction des angles de torsion adoptés autour de leurs liaisons chimiques. L'angle ω est en fait le nom donné à celui de la liaison inter-résidu *C-N*. Ce lien à un caractère partiel de double liaison, ce qui limite à deux les valeurs prises par l'angle ω : 0° et 180°.

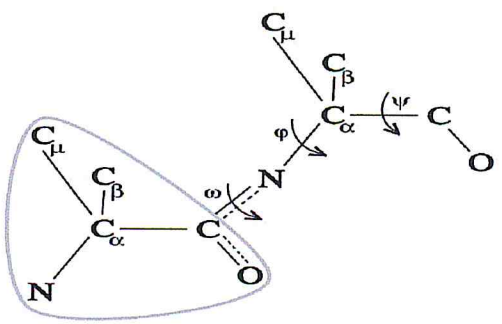


Fig.1.5 Illustration de l'angleω . La molécule entourée est un résidu quelconque, venu s'attacher au résidu que nous étudions.

6.2 *Le modèle HP* [AG08]

Le modèle HP de *Chan et Dill,* développé en 1993, est une représentation très simpliste d'une protéine, illustré à la ***(figure1.6).***

Il propose de séparer les résidus en deux classes : les résidus hydrophiles notés P et les résidus hydrophobes notés H. Chaque résidu est représenté par un seul atome virtuel, possédant comme seule caractéristique propre la propriété hydrophile ou hydrophobe.

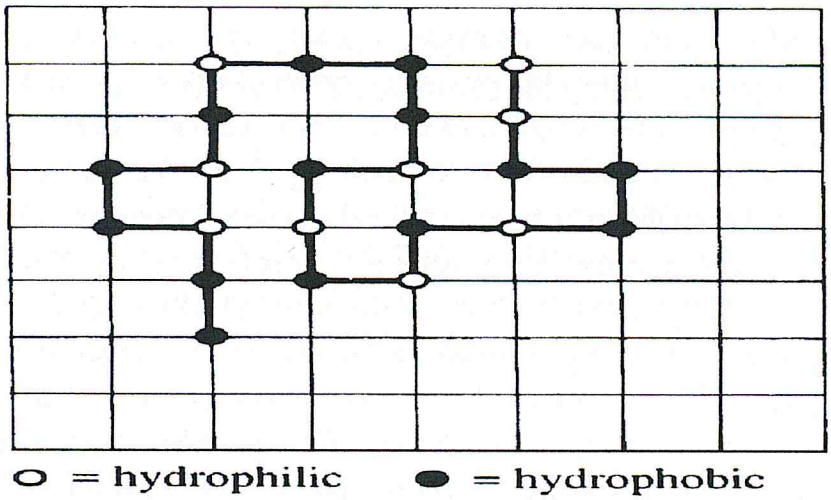


Fig.1.6 Modèle HP de Chan et Dill, 1993[MBO08]

Ce modèle étant basé sur la théorie de la création d'un noyau hydrophobe, il faudra, pour satisfaire aux conditions de repliement des protéines, minimiser l'interaction des résidus *H* avec l'environnement extérieur (généralement l'eau).

Le résultat d'une prédiction effectuée à partir d'un modèle de ce genre se trouve en ***(figure 1.7)***

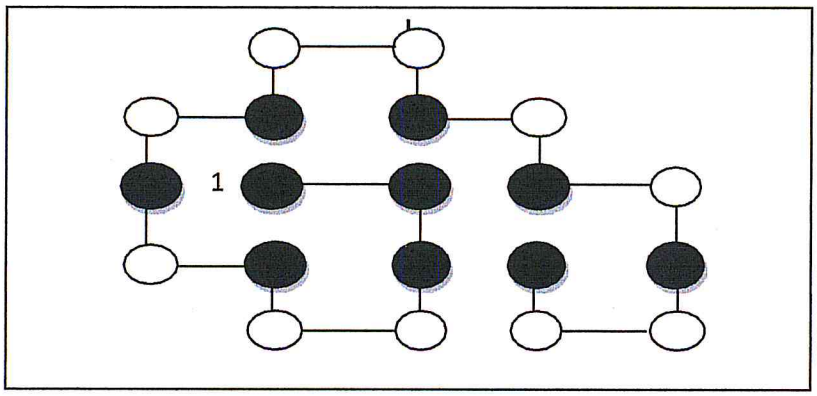


Fig. 1.7 Résultats d'une prédiction bidimensionnelle basée sur le modèle HP pour une protéine de séquence 'HHHPPHPHPHPPHPHPHPPH' (le début de la séquence est symbolisée par '1' et les résidus 'H' sont en noir). **[AG08]**

6.3 *Le modèle intermédiaire* [AG08]

Offre une description moins détaillé que le modèle (φ,ψ,ω) et plus détaillée de la protéine que le modèle H-P en regroupent les acides aminés en plusieurs classes afin de permettre un niveau de détaille plus élevé.

7. Conclusion

Le repliement de protéine est un processus extrêmement complexe, il fait l'objet de plusieurs études.

L'état natif correspond à l'énergie minimale. La prédiction de cette conformation de moindre énergie peut être réalisée en se basant seulement sur la séquence de la structure primaire.

Plusieurs modèles ont été élaborés pour cette méthode, et malgré leur simplification, la résolution du problème sur ces modèles d'une manière exacte reste impossible (problème NP-Complet, explosion exponentielle).

Pour cela il existe plusieurs outils permettant la résolution de ce problème d'une façon approché, c'est le contenue du chapitre suivant.

CHAPITRE 2

OPTIMISATION COMBINATOIRE

&

MÉTHODES DE RÉSOLUTION

CHAPITRE2

OPTIMISATION COMBINATOIRE

&

MÉTHODES DE RÉSOLUTION

I. Introduction

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 valable pour toutes les données **[JKH99].**

Un problème d'**optimisation combinatoire** consiste à trouver la meilleure solution dans un ensemble discret de solutions appelé ensemble des solutions réalisables.

En général, cet ensemble est fini mais de cardinalité très grande et il est décrit de manière implicite, c'est à dire par une liste de contraintes que doivent satisfaire les solutions réalisables.

Pour définir la notion de **meilleure solution**, une fonction, dite fonction objectif, est introduite. Pour chaque solution, elle renvoie un réel et la meilleure solution est celle qui minimise ou maximise la fonction objective. **[CEDRIC]**

Ce chapitre a pour but de décrire certaine bases théorique et de comprendre les notions de la complexité algorithmique et des problèmes d'optimisation combinatoire et les différents méthodes de résolutions.

II. Définition de l'Optimisation combinatoire

On qualifie généralement de « combinatoires » les problèmes dont la résolution se heurte à une explosion du nombre de combinaisons à explorer (nombre de solution très grand) **[RCS]**

Etant donné une fonction $f: A \rightarrow R$ d'un ensemble A dans l'ensemble des nombre réels.
Rechercher un élément x0 de A tel que $f(x0) \geq f(x)$ pour tous les x en A (« maximisation ») ou tel que $f(x0) \leq f(x)$ pour tous les x en A (« minimisation »).
Les éléments de A sont appelés les « solutions admissibles » et la fonction f est appelée « la fonction objectif »
Une solution possible qui maximise (minimise) la fonction objectif est appelée une « solution optimale ». **[OP08]**

Cette notion de problème combinatoire est formellement caractérisée par la théorie de la complexité qui propose une classification des problèmes en fonction de la complexité de leur résolution.

III. Complexité algorithmique et optimisation combinatoire

La théorie de la complexité s'intéresse à l'étude formelle de la difficulté des problèmes en informatique, elle se concentre donc sur les problèmes qui peuvent effectivement être résolus.

La théorie de complexité met en évidence deux types de problème : les problèmes de *décision* et les problèmes *d'optimisation combinatoire.*

- ***Un problème de décision*** est un problème dont les résultats ne peuvent prendre que l'une des deux valeurs: VRAI ou FAUX.
- ***Un problème d'optimisation combinatoire*** est un problème d'optimisation d'une fonction (maximisation/minimisation), dite fonction objective, définie sur un ensemble de *solution fini,* dit espace de recherche, qui satisfait des contraintes. **[MBO08]**

Ces deux types de problèmes sont liés par le concept d'association. A chaque problème d'optimisation combinatoire, il correspond un problème de décision. Les problèmes d'optimisation, peuvent être facilement transformés en un problème de décision équivalent.

Le but de la théorie de la complexité est la classification des problèmes de décision suivant leur degré de difficulté de résolution. Elle répons aux questions du type « est-ce qu'on peut résoudre un tel problème ? » .Dans la littérature, il existe plusieurs classes de complexité, sont les suivantes :

- ***La classe P*** c'est la classe des problèmes pouvant être résolus en un temps polynomial, dits facile. **[JR08]**

- ***La classe NP*** c'est l'abréviation pour Non-deterministic Polynomial time est la classe de tous les problèmes de décision dont la solution peut être vérifiée en temps polynomial, *i.e.* si l'on nous donne une solution certifiée, il est possible de vérifier que cette solution est correcte en un temps polynomial par rapport à la taille de l'entrée. **[JR08]**

  Un problème NP est NP-complet si tout problème NP s'y réduit en temps polynomial. Autrement, les problèmes NP les plus difficiles sont NP-Complets.

- ***La classe des problèmes NP-Difficile*** Un problème NP-Difficile est un problème d'optimisation dont le problème de décision associé est NP-Complet., On a NP-complet appartiens a NP-difficile, par contre un problème NP-difficile n'est pas nécessairement dans NP. **[JR08]**

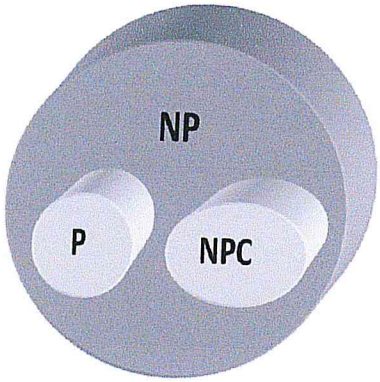


Fig.2.1 Relations entre les classes P, NP et NP-Complet **[MBO08]**

IV. Les méthodes de résolution

Pour la résolution des problèmes d'optimisation qui sont pour la plupart NP-difficile, des méthodes exactes, ainsi que des heuristique on été proposés

- **Les approches exactes [ACO08]**

Fournissent la solution optimale d'un problème donné, mais peuvent s'avérer très couteuse en temps et en mémoire. Elles sont efficace pour des instances de problèmes de petites taille, étant donnée la nature des problèmes que nous traitions, il est peu réaliste d'utiliser ces méthodes.

- **Méthodes approchées [ACO08]**

Constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille, ces méthodes sont utilisées depuis longtemps par de nombreux praticiens.

Pour les grands problèmes NP-durs, il est même probablement impossible de trouver une solution optimale durant une durée de vie humaine, les méthodes approchée permettent d'obtenir des résultats approchés plus rapidement et en un temps raisonnable.

La **(figure 2.2)** donne une des classifications possibles de ces méthodes. C'est celle qui sera adoptée pour la suite du chapitre.

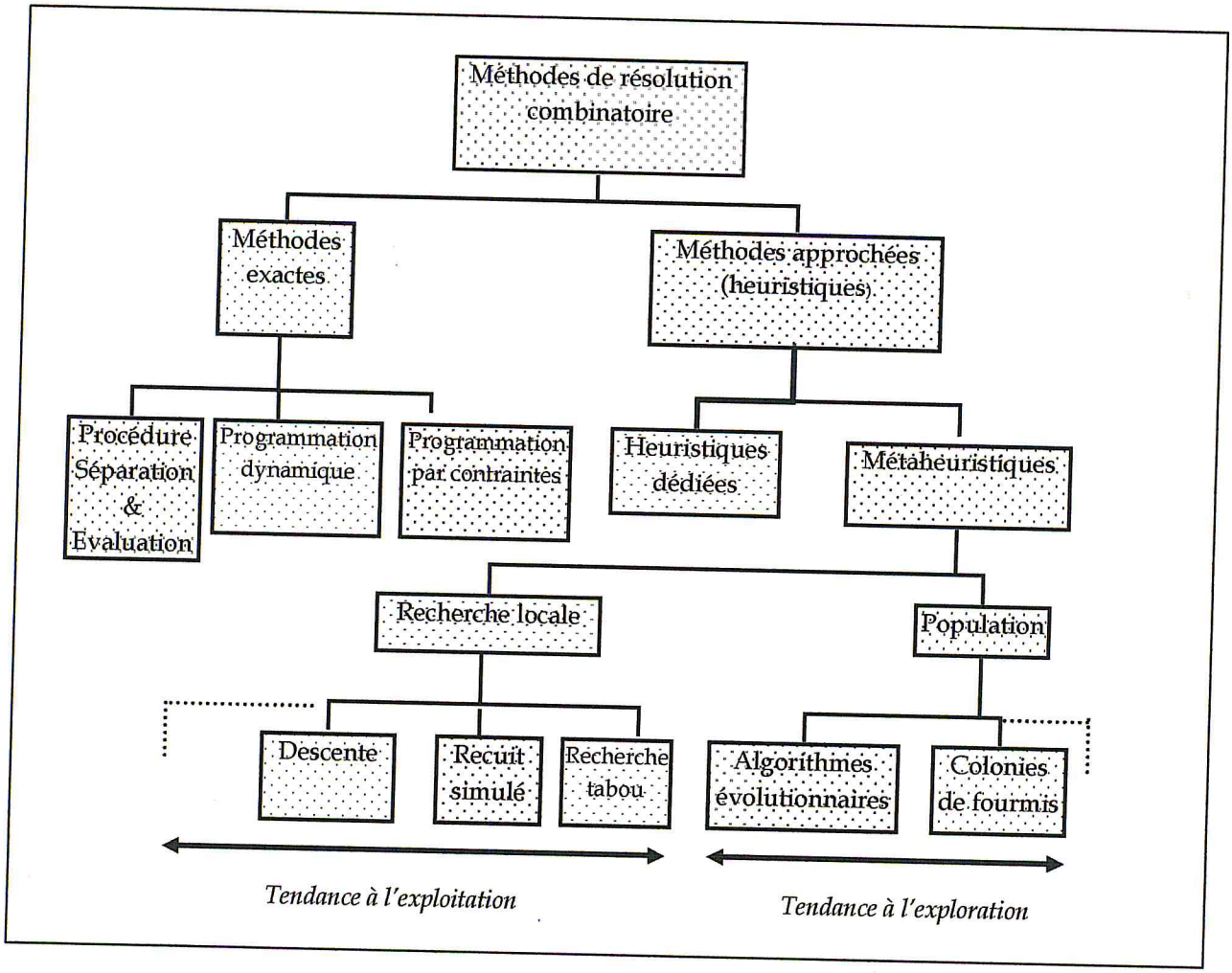


Fig.2.2 Classifications des méthodes de résolution combinatoire **[ACO08]**

V. Méthodes approchées (les heuristiques)

A. Les heuristiques dédiées

Les heuristiques dédiées, ou spécifiques, appartiennent à la catégorie des méthodes approchées. Une heuristiques dédiée est un algorithme de résolution ne fournissent pas nécessairement une solution optimale pour un problème d'optimisation donné mais plutôt acceptable. **[JKH99]** .Cette définition sous entend

qu'une heuristique dépend du problème à résoudre et ne peut donc pas être généralisée à un ensemble de problèmes.

Une bonne heuristique possède cependant plusieurs caractéristiques :

- Elle est de complexité raisonnable (idéalement polynomiale mais en tout cas efficace en pratique) ;
- Elle fournit le plus souvent une solution proche de l'optimum ;
- La probabilité d'obtenir une solution de mauvaise qualité est faible ;
- Elle est simple à mettre en œuvre. **[HEC03].**

Comme les heuristiques ne garantissent pas l'atteinte d'un optimum global, leur utilisation est justifiée dans certains cas **[ZAN89]**, soit :

1. Les données sont inexactes ou limitées.
2. Un modèle simplifié du problème est utilisé.
3. Aucune méthode exacte fiable n'existe.
4. Une méthode exacte existe, mais n'est pas intéressante au niveau informatique parce que trop vorace en temps de calcul ou en espace mémoire.
5. Elles sont utilisées pour améliorer la performance d'un optimiseur.
6. Il est requis de résoudre le même problème fréquemment.
7. Une solution heuristique est acceptable.
8. On cherche une méthode simple et pouvant être comprise par les utilisateurs.
9. Elles servent de dispositifs d'apprentissage.

Et/ou

10. Lorsqu'il existe d'autres limites concernant les ressources.

B. Les métaheuristiques

Apparues dans les années 80,c'est un mot grec, il est composé dans un premier temps du préfixe « méta » qui signifie « au-delà» ou « plus haut » puis de « heuristique » qui signifie « trouver »,cette décomposition permet facilement de

comprendre le but premier de ces algorithmes, c'est de trouver des solutions a des problèmes en utilisons plusieurs heuristiques.

Les métaheuristiques ont changé radicalement l'élaboration d'heuristiques. Ces dernières commençaient par s'interroger sur les caractéristiques et les particularités du problème à résoudre avant de commencer à programmer une méthode spécifique. Avec les métaheuristiques on inverse en quelque sorte ce processus. Ayant une première heuristique, on cherche ensuite à l'améliorer en observant les faiblesses qu'elle présente pour le problème à résoudre. **[TAI02].**

Notons qu'une métaheuristiques n'est pas propre à un problème précis, mais adaptable et applicable à une large classe de problèmes.

Les métaheuristiques peuvent être classées en deux grandes familles : *la recherche locale*, qui tend plus vers la notion de voisinage, et *les métaheuristiques à population* qui, quand à eux, tendent vers l'exploration de domaine de recherche.

1. Les métaheuristique à population

Appelé aussi « Les méthodes évolutives », représentent une classe de métaheuristiques qui sont basés sur le principe du processus d'évolution naturelle, ils doivent leur nom à l'analogie avec les mécanismes d'évolution des espèces vivantes. Un algorithme évolutif se compose de la population, un mécanisme d'évaluation, et un mécanisme d'évolution composé d'opérateurs. **[DAP08]**

On peut distinguer dans cette catégorie les algorithmes génétiques, et les colonies de fourmis.

A. Les algorithmes génétiques [BA09]

Se sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l''evolution naturelle: croisements, mutations, sélection, etc. Les algorithmes génétiques ont déjà une histoire relativement ancienne puisque les premiers travaux de John Holland sur les systèmes adaptatifs remontent à 1962. L'ouvrage de David Goldberg [Gol89c] a largement contribué à les vulgariser.

L'algorithme fait appel à quatre opérateurs de base :

L'évaluation, La sélection, Le croissement et **La mutation**

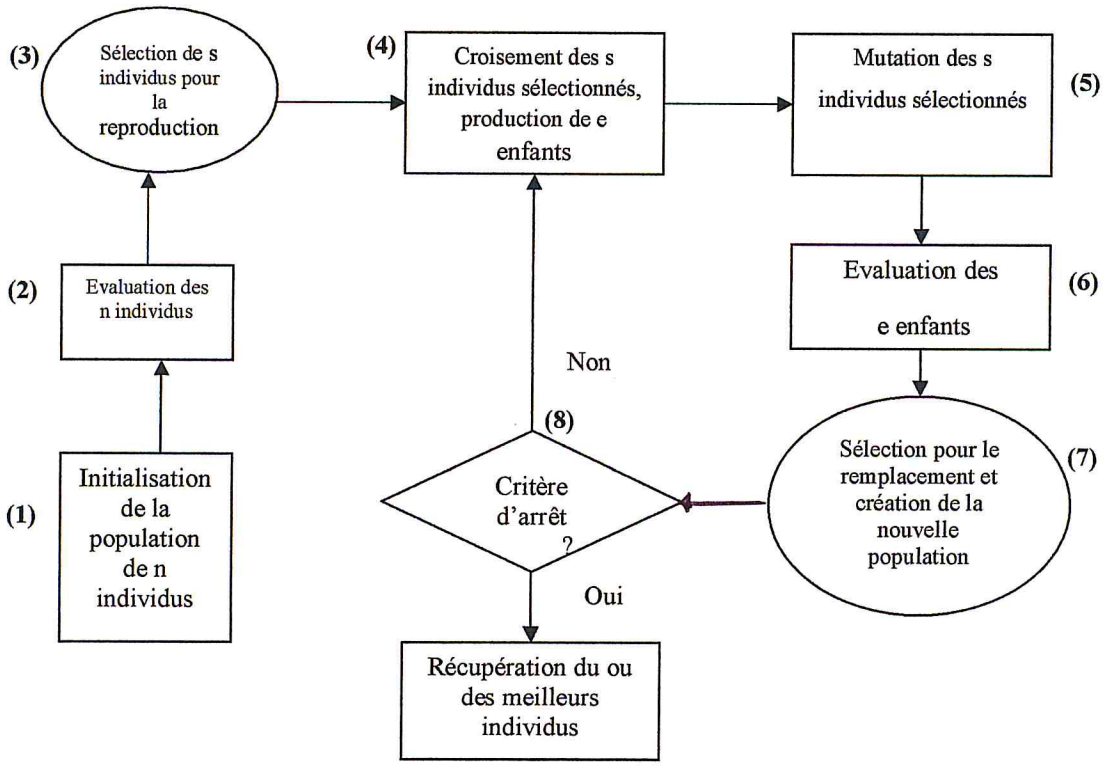


Fig.2.3.Organigramme de la métaheuristique génétique **[AG05]**

Explication de l'organigramme :

(1) *Initialisation* : la population est générée aléatoirement. La diversité génétique est un facteur de l'accélération de la convergence et une condition au bon fonctionnement de la métaheuristique.

(2) *Évaluation* : On déterminer le phénotype de chaque individu de la populatio n.

(3) *Sélection* : on sélectionne des individus pour la reproduction

(4) *Croisement* : L'algorithme opère un croisement de ces individus. En règle générale, seules les meilleures solutions sont gardées pour la reproduction.

(5) *Mutation :* Les individus sélectionnés mutent. Le but de cette étape est de pouvoir définir de nouveaux espaces de recherches locaux en partant de nouvelles directions de mutations.

(6) *Évaluation :* on évalue les enfants issues des reproductions et mutations afin de pouvoir comparer leur qualité.

(7) *Remplacement :* enfin on sélectionne les meilleurs enfants pour effectuer un remplacement générationnel sur les anciens individus et ainsi créer une nouvelle population.

(8) *Arrêt :* si le critère d'arrêt est atteint on arrête la métaheuristique sinon on recommence à partir de la phase de Sélection pour une itération (Sélection→ Croisement →Mutation → Evaluation) avec la nouvelle population. Ce critère d'arrêt peut être un temps maximal, un nombre de population limite ou un test sur le meilleur individu (c.-à-d. la meilleure solution obtenue).

B. Les Colonies de fourmis

L'optimisation par colonie de fourmis ACO pour "Ant Colony Optimization" est une métaheuristique relativement récente qui s'inspire de l'intelligence collective des fourmis. Initialement, elle a été conçue pour résoudre un problème discret classé NP-difficile.

Relativement récente, elle a été introduite en 1991 par *Colorni Dorigo* et *Maniezzo* pour résoudre le problème du Voyageur de commerce. Elle s'est popularisée, puis a été l'objet d'améliorations dès 1995 et a été appliquée avec succès à d'autres problèmes d'optimisation combinatoire dès 1994 **[Dor97]**. Nous allons tout d'abord exposer les points communs entre les fourmis virtuelles et les fourmis réelles, avant d'exposer la métaheuristique. Ceci expliquera comment les fourmis virtuelles peuvent être exploitées pour résoudre un problème d'optimisation combinatoire.

a. Relation avec l'informatique

En observant une colonie de fourmis à la recherche de nourriture dans les environs du nid, on s'aperçoit qu'elle résous des problèmes tels que celui de la recherche du plus court chemin. Les fourmis résolvent des problèmes complexes par

des mécanismes assez simples à modéliser. Il est ainsi assez simple de simuler leur comportement par des algorithmes.

b. Comportement de la fourmi

En marchant du nid à la source de nourriture (ce qui dans un premier temps se fait essentiellement de façon aléatoire), les fourmis déposent au passage sur le sol une substance odorante appelée phéromones. Cette substance permet ainsi donc de créer une piste chimique, sur laquelle les fourmis s'y retrouvent. En effet, d'autres fourmis peuvent détecter les phéromones grâce à des capteurs sur leurs antennes.

Les phéromones ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent être utilisées par les autres fourmis pour retrouver les sources de nourritures trouvées par leurs congénères.

Ce comportement permet de trouver le chemin le plus court vers la nourriture lorsque les pistes de phéromones sont utilisées par la colonie entière. Autrement dit, lorsque plusieurs chemins marqués sont à la disposition d'une fourmi, cette dernière prend le chemin le plus court vers sa destination. **[OPF06]**

-Nous exposons quelques expériences pour illustrer d'avantage le comportement coopératif des fourmis.

- **Expérience du pont binaire**

L'expérience montre un nid d'une colonie de fourmis, qui est séparé d'une source de nourriture par un pont à deux voies de même longueur. On laisse évoluer les fourmis sur le pont, on trace ainsi en fonction du temps, le graphe du nombre de fourmis empruntant chaque branche. Le résultat de l'expérience est exposé à la (**figure 3.1**). L'illustration (a) représente la configuration physique de l'expérience. Le graphique (b) indique l'évolution de ce système en fonction du temps : on constate que les fourmis ont tendance à emprunter le même chemin (par exemple celui du haut) après une dizaine de minute.

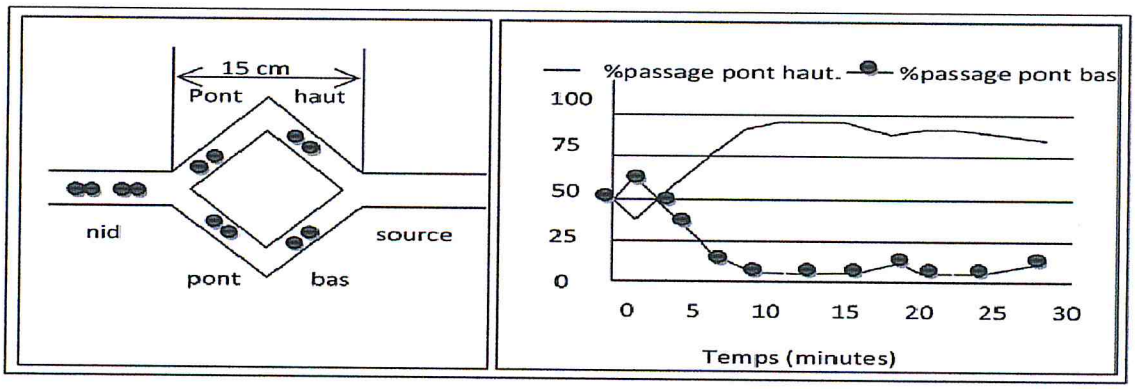


Fig.2.4Expérience du pont binaire

Au départ, il n'y a pas de phéromone sur le pont. Donc, chaque branche peut être choisie par une fourmi avec la même probabilité. Néanmoins, dans notre exemple, après une certaine période, des variations aléatoires ont fait qu'un peu plus de fourmis ont choisi le chemin du haut plutôt que celui du bas. Puisque les fourmis déposent des phéromones en avançant et puisqu'elles sont plus nombreuses en haut qu'en bas, le chemin du haut comportera plus de phéromones. Cette quantité supérieure de phéromone incite plus de fourmis à choisir la branche du haut, donc la quantité de phéromone déposée augmentera encore plus. On en déduit que plus les fourmis suivent un chemin, plus ce chemin devient intéressant à suivre. Ainsi la probabilité avec laquelle une fourmi choisit un chemin, augmente avec le nombre de fourmis qui ont pris ce chemin précédemment.

- **Expérience du double pont binaire**

On peut se demander à présent quel serait l'effet de l'augmentation de la longueur d'une des deux branches du pont. L'effet produit sera que la branche la plus courte sera sélectionnée.

En effet, les premières fourmis qui reviennent au nid avec de la nourriture sont celles qui ont emprunté le chemin le plus court dans les deux sens. Ce chemin, marqué deux fois par les phéromones, attire plus les autres fourmis que le long chemin, qui lui est marqué une seule fois dans le sens de l'aller. Cet effet se renforce au fur à

mesure, jusqu'à ce que toutes les fourmis choisissent le chemin le plus court. C'est ainsi que dans cette expérience, on voit que les variations aléatoires sont réduites, puisque les deux chemins n'ont plus la même longueur. Contrairement à la première expérience, le comportement des fourmis qui consistait à suivre les pistes de phéromones n'est plus le seul mécanisme présent, maintenant on associe ce mécanisme à une notion de distance.

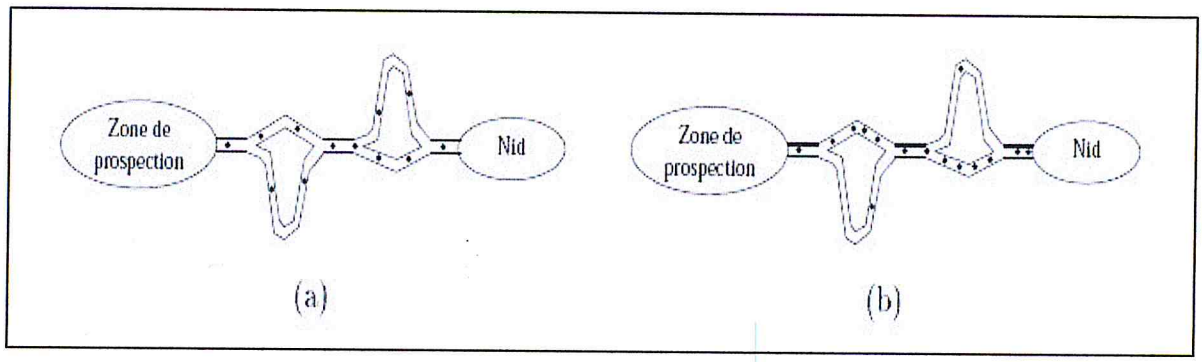


Fig. 2.5 Expérience du double pont binaire

Toutefois, quand le chemin le plus court n'est ouvert qu'après le chemin plus long (par exemple, quand le chemin était au départ bloqué par un obstacle), les fourmis continuent de parcourir le chemin le plus long car c'est le seul à être recouvert de phéromone. C'est ainsi que , l'évaporation des phéromones joue un rôle capital : les pistes de phéromones sont moins présentes sur les chemins les plus longs, car le réapprovisionnement en phéromone demande plus de temps que sur les chemins plus courts. Donc il suffit alors qu'une poignée de fourmis choisissent le chemin auparavant bloqué pour favoriser celui-ci. Ainsi, même quand des voies plus courtes ont été découvertes après, les fourmis finissent par les emprunter.
On peut généraliser cela à plus de deux chemins possibles : dans la**(figure 2.5 (a)),** on a utilisé un double pont avec quatre chemins possibles de différentes longueurs. On voit dans le dessin **(b)** que la plupart des fourmis finissent par choisir le chemin le plus court. Les expériences montrent que quand environ cent fourmis ont déjà emprunté le pont, plus de 90% d'entre elles sélectionnent le chemin le plus court : les fourmis convergent donc assez rapidement.

- **Expérience de l'obstacle**

Montre que les fourmis sont capables d'effectuer des détournements, si un obstacle a été placé sur leur chemin entre la source de nourriture et le nid, pour retrouver le plus court chemin **[Dor97]**.

Lorsqu'un obstacle est posé sur le chemin des fourmis, elles choisissent d'aller à gauche ou à droite avec des probabilités égales (**Fig2.6 (b)**). Le chemin le plus court sera le plus rapidement parcouru par les fourmis (**Fig2. 6 (c)**). En conséquence, la plus grande quantité de phéromone est posée rapidement sur le plus court chemin, et puisque les fourmis choisissent le trajet qui contient le plus de phéromone, le plus court chemin sera utilisé.

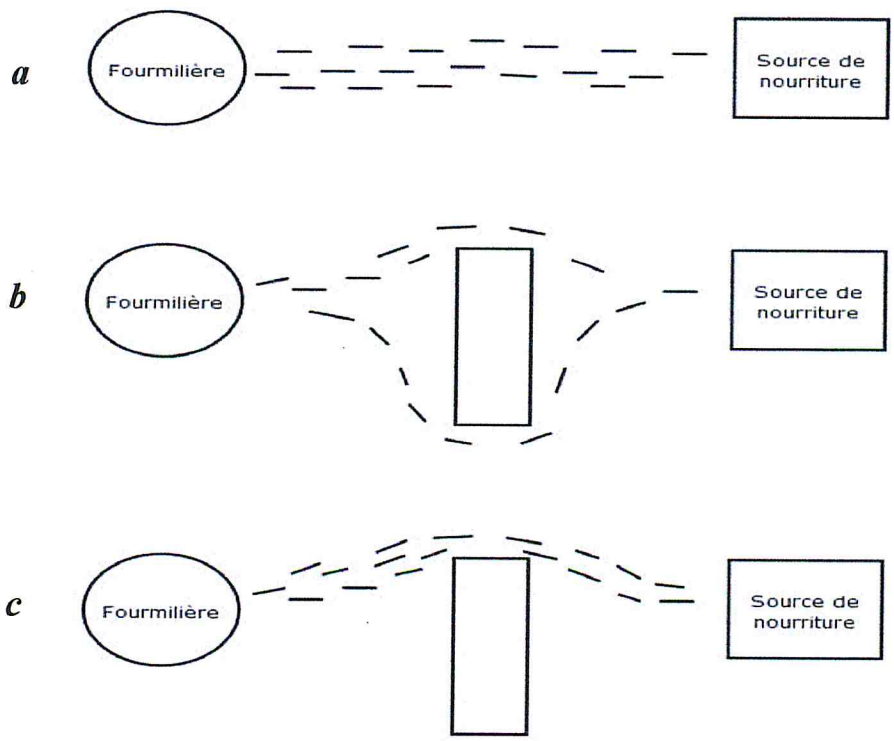


Fig.2.6 Expérience de l'obstacle

En s'appuyant sur ces expériences, la métaheuristique de l'optimisation par colonies de fourmis est apparue.

c. *Caractéristiques d'une colonie de fourmis [RRC07]*

Une colonie de fourmis est caractérisée par :

- *Le parallélisme* : Dans une colonie de fourmis, plusieurs fourmis travaillent en même temps pour trouver une solution au problème (i.e. : trouver la nourriture).
- *La flexibilité* : Une colonie de fourmis est capable de s'adapter à des modifications de l'environnement (elle est capable de détourner les obstacles par exemple).
- *La robustesse* : Une colonie de fourmis est apte à continuer son travail même si certains individus sont défaillants ou échouent.
- *La décentralisation* : Une colonie de fourmis n'obéit pas à une autorité centralisée (Il n'y a pas de hiérarchie et pas de reine qui dicte le travail à faire). Elle est de ce fait qualifiée d'Hétérarchie dense ou de réseau hautement connecté.
- *L'auto-organisation* : Une colonie de fourmis trouve d'elle-même une solution qui n'est pas connue à l'avance.

d. *Les fourmis artificielles* [OPF06]

Les fourmis virtuelles ont une double nature. D'une part, elles modélisent les comportements abstraits de fourmis réelles, et d'autre part, elles peuvent être enrichies par des capacités que ne possèdent pas les fourmis réelles, afin de les rendre plus efficaces que ces dernières. Nous allons synthétiser ces ressemblances.

- Points communs

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

- Pistes de phéromones: Ces fourmis communiquent par le mécanisme des pistes de phéromone. Cette forme de communication joue un grand rôle dans le comportement des fourmis : son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis, en fonction de l'historique laissé par ces phéromones.
- Évaporation des phéromones. La métaheuristique ACO comprend aussi la possibilité d'évaporation des phéromones. Ce mécanisme permet d'oublier lentement ce qui s'est passé avant. C'est ainsi qu'elle peut diriger sa recherche vers de nouvelles directions, sans être trop contrainte
- Recherche du plus petit chemin: Les fourmis réelles et virtuelles partagent un but commun recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).
- Déplacement locaux Les vraies fourmis ne sautent pas des cases, tout comme les fourmis virtuelles. Elles se contentent de se déplacer entre sites adjacents du terrain.
- Choix aléatoire lors des transitions. Lorsqu'elles sont sur un site, les fourmis réelles et virtuelles doivent décider sur quel site adjacent se déplacer. Cette prise de décision se fait au hasard et dépend de l'information locale déposée sur le site courant. Elle doit tenir compte des pistes de phéromones, mais aussi du contexte de départ (ce qui revient à prendre en considération les données du problème d'optimisation combinatoire pour une fourmi virtuelle).

Caractéristique des fourmis artificielle **[DOC07]**

Marco Dorigo est à l'origine de l'OCF. Il estime que rechercher une source de nourriture est analogue à rechercher une solution dans un espace de recherche commun. L'OCF s'appuie sur des agents appelés fourmis artificielles ayant des capacités supérieures. De ce fait, les fourmis artificielles se caractérisent par :

- Elles évoluent d'un état à un autre dans monde discret.
- A chaque fourmi est associée une mémoire qui contient l'historique de ses actions.
- La quantité de phéromones déposée par les fourmis réelles représente une valeur stockée dans une variable pour les fourmis artificielles et

dépend du problème à traiter. L'évaporation des phéromones est une décrémentation de la valeur.

e. Optimisation par les colonies de Fourmies (ACO)

L'optimisation par colonie de fourmis, dénommée ACO pour Ant Colony Optimisation en anglais, est une métaheuristique généralisée proposée par Marco Dorigo. Elle est destinée aux problèmes d'optimisation combinatoire. Une formalisation de l'algorithme de colonie de fourmis donne une représentation du problème, décrit le comportement des fourmis artificielles dans la représentation et fait part de l'organisation générale de la métaheuristique. Les propriétés des fourmis ainsi que l'algorithme décrivant la métaheuristique sont abordés plus loin.

(a) Formalisation

- ***Représentation du problème à résoudre*** **[RRC07]**

Considérons la minimisation du problème d'optimisation combinatoire (S, f, Ω), avec S l'ensemble des solutions candidates, f la fonction objective qui assigne à chaque solution $s \in S$ un *cout* $f(s,t)$ (t si on prend en considération le temps), et Ω un ensemble de contraintes. Le but est de trouver un optimum globale $s_{opt} \in S$, qui est la solution de *coût* minimum qui satisfait les contraintes Ω.

Le problème (S, f, Ω) est modélisables par un graphe G = (C, L) tel que :

Le graphe est caractérisé par :

- Un ensemble fini de composants C= $\{c_1, c_2, \ldots\ldots c_{Nc}\}$ où les c_i sont appelés sommets (nœuds) et Nc= Card(C).
- Un ensemble fini de connexions (transitions ou arcs) L= $\{l_{i\ j} / (c_i, c_j) \in C \times C\}$ où $l_{i\ j}$ est le passage du sommet i vers le sommet j en un seul saut ou un pas logique.
- Un coût $J_{i\ j}$ associé à chaque connexion $l_{ij} \in L$, qui peut être parfois paramétré par le temps J_{ij} (t).
- Un ensemble fini de contraintes $\Omega \equiv \Omega$ (C, L, t) dans le cas où c'est paramétré.

- Une séquence $s = < c_i, c_j, \ldots\ldots c_k, \ldots\ldots >$ de sommets (ou une séquence $s = < l_{ij}, l_{kl}, \ldots\ldots l_{mn}, \ldots\ldots >$ de transitions). La séquence s est appelée état du problème. Si S est l'ensemble de toutes les séquences possibles, Ś est l'ensemble de toutes les séquences ou sous-séquences qui respectent les contraintes Ω (C, L, t). Ceci implique que Ś est inclus ou égal à S. Autrement dit, les éléments de S définissent les états possibles du problème.
- La structure de voisinage : étant donné deux états s_1 et s_2, on dit que s_2 est un voisin de s_1, si s_1 et s_2 appartiennent à S, et on peut atteindre s_2 à partir de s_1 en un seul pas logique (si c_1 est le dernier composant de la séquence s_1, il existe c_2 ϵ C, tel que $l_{ij} \epsilon L$ et $s_2 \equiv < s_1, c_2 >$).
- Ψ est une solution si elle appartient à Ś et satisfait toutes les contraintes du problème.
- J_Ψ (L, t) est le coût associé à chaque solution Ψ. J_Ψ (L, t) est en fonction de tous les coûts J_{ij} des connexions appartenant à la solution Ψ.

- Comportement [RRC07]

L'algorithme des colonies de fourmis est le résultat d'un travail collectif établi par des fourmis artificielles. Ces dernières produisent des solutions pas toujours optimales en démarrant d'un sommet et en arrivant à un autre tout en passant par d'autres nœuds.

Le choix du prochain nœud à atteindre (ou du prochain arc à emprunter) se fait suivant une règle de décision probabiliste fonction des pistes de phéromone locales, de l'état de la fourmi et des contraintes du problème. Une fois la solution construite, les fourmis artificielles déposent une quantité τ_{ij} de phéromone sur les arcs d'après la qualité de la solution.

L'algorithme est itéré et les quantités de phéromone déposées vont guider les fourmis dans les prochaines itérations. L'algorithme s'arrête une fois la condition d'arrêt satisfaite ou jusqu'à une convergence des solutions trouvées par les fourmis. Une table de phéromone est utilisée pour guider les fourmis dans leur recherche. Egalement, des valeurs heuristiques η_{ij} représentant des contraintes du problème peuvent être associées aux arcs l_{ij}.

(b) La colonie et ses fourmis [DOR99]

Pour aboutir à de bons résultats, les fourmis de la colonie coopèrent entre elles en échangeant des informations sur l'état actuel du problème. Dans ce qui suit nous allons citer les caractéristiques de la colonie entière et celles propres aux fourmis en particulier.

- Les propriétés de la colonie de fourmis

La colonie de fourmis est caractérisée par les propriétés suivantes :

- Chaque fourmi cherche à trouver une solution propre à elle.
- Chaque fourmi utilise ses propres informations et les informations locales des nœuds (composants) visités.
- Une fourmi communique avec les autres indirectement, à travers les informations lues/écrites de variables sauvegardant les valeurs de phéromone.

- Propriétés des fourmis

En résumé, chaque fourmi de la colonie possède les propriétés suivantes :

- Une fourmi cherche une solution réalisable à coût minimal.
- Une fourmi k possède une mémoire M^k, elle est utile pour :
  - Construire une solution réalisable.
  - Evaluer une solution trouvée.
  - Retracer le chemin trouvé.

- Les fourmis construisent leur solution d'une manière progressive. Elles démarrent à partir de l'état initial et passent par les voisins possibles. La procédure de construction se termine quand toutes les fourmis vérifient leurs critères d'arrêt.

- Une fourmi k se trouvant sur un nœud i peut aller vers un nœud j de son voisinage en utilisant une règle de décision probabiliste.

(c) Algorithme générale d'ACO [RRC07] [ACO08]

L'algorithme ACO est donné dans ce qui suit:

Algorithme colonie de fourmis ( )

Début

Initialisation

Tant que (condition _d'arrêt_ non_ satisfaite) ***faire***

Pour chaque fourmi ***faire***

Choisir un nœud en appliquant la règle de transition d'état.

Mise à jour de la liste taboue

Mise à jour locale

Fait

Fait

Mise à jour de la meilleure solution

Appliquer la mise à jour de phéromone (mise a jour globale)

Fin.

Les caractéristiques de l'algorithme ACO

- **Les pistes de phéromone [ACO08]**

Le premier point le plus important quand on applique l'algorithme d'ACO est *la définition de la signification des pistes de phéromone.* En effet, le choix de cette définition est en grande partie liée aux possibilités de représentation de l'espace de recherche, chaque représentation pouvant apporter une façon différente

- Peuvent être calculées une fois à l'initialisation.
- A chaque itération de l'algorithme ACO, une table peut être pré-calculée avec les valeurs de phéromone.

b) Heuristique dynamique

Dans le cas des problèmes dynamiques les informations heuristiques sont calculées aux cours de traitement. Elles dépendent de la solution partielle construite. Par conséquent elles doivent être recalculées à chaque pas de la promenade d'une fourmi. Cela engendre un temps de calcul plus élevé.

▪ Le nombre de fourmis [RRC08]

Pourquoi utiliser une colonie de fourmis au lieu d'utiliser une seule fourmi?

Bien qu'une fourmi seule soit capable de produire une solution, mais de qualité moindre, il est plus intéressant d'utiliser toute une colonie. Dans le cas de problème d'optimisation combinatoire, l'usage de m fourmis, m>1 qui construisent r solutions chacune (c.-à-d., l'algorithme ACO est exécuté r fois) pourrait être équivalent à l'usage d'une seule fourmi qui produit $m*r$ solutions. Les résultats théoriques sur la convergence des algorithmes ACO montrent qu'ils sont performants quand le nombre de fourmis m est supérieur à 1, étant donné qu'une seule fourmi peut ne pas bien explorer l'espace de recherche. Les solutions de bonne qualité sont le résultat de l'interaction collective des fourmis. Elles sont obtenues par la communication indirecte en lisant et écrivant des informations dans les variables sauvegardant les valeurs de phéromone.

▪ 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 rassemblée 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 l'aléatoire » dans le système (*diversification*) et/ou améliorer une solution (*intensification*).

Dans les algorithmes de type ACO, il existe plusieurs façons de gérer l'emploi de ces deux facettes des métaheuristique 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. A l'inverse, plus α sera faible, plus la *diversification* sera forte, car les fourmis éviteront les pistes. Le paramètre β agit de façon similaire. On doit donc gérer à la fois les deux paramètres pour régler ces aspects **[ACO 08]**.

(d) Les effets de l'algorithme ACO [HAM03]

L'algorithme ACO peut inclure deux effets importants :

- ***L'évaporation de phéromone***

L'intensité de la phéromone est décrémentée dans le temps, affin de rendre les arcs visités de moins en moins attirants. En favorisant l'exploration des chemins les moins visités, les fourmis auront tendance à ne pas converger vers un chemin. Si les fourmis explorent des chemins différents, alors il y a une plus forte probabilité que l'une d'elles trouve une amélioration de la solution. Par contre, si elles convergent toutes vers le même chemin, l'utilisation de m fourmis sera inutile.

- ***Les actions de démon***

Les actions de démons (processus de veille) peuvent être utilisées pour implémenter des actions centralisées, qui ne peuvent pas être effectuées par les fourmis individuelles. Le démon peut observer le chemin trouvé par chaque fourmi de la colonie, et rajouter de l'extra phéromone sur les composants utilisés par la fourmi qui a construit le meilleur chemin (solution). Ceci en respectant la phéromone "on-line" déjà déposée. Cette mise à jour est dite la mise à jour "offline" de phéromone.

(e) Domaines d'applications de colonies de fourmis [SCFR09]

Les applications des algorithmes de colonie de fourmis sont de plus en plus nombreuses, et de plus en plus évoluées.

- **le problème de voyageur de commerce (PVC)**

Le PVC est l'un des premiers problèmes à avoir suscité l'utilisation de la méthodologie des colonies de fourmis pour résoudre le problème.

Un voyageur de commerce doit visiter un ensemble $\{c_1, c_{2,...}, c_n\}$ de villes dont on connait les distances respectives. Le problème consiste à trouver le chemin de longueur minimal qui passe une et une seul fois par chacune des villes et qui revient finalement à la ville de départ.

- **Les problèmes de satisfiabilité SAT, MAX-SAT et MAX-W-SAT**

Le problème de satisfiabilité, SAT en abrégé est un parmi les problèmes les plus connus en intelligence artificielle et en théorie de la complexité. Il consiste à trouver une interprétation (instanciation) qui peut mettre à vrai toutes les clauses d'une formule booléenne écrite sous forme normal conjonctive (FNC). On dit alors que cette dernière est satisfiable. A titre d'exemple, la formule F écrite sous forme normal conjonctive F= (a v b) ^ (av c) ^ (a), est satisfiable pour (0, 1,0) et (0, 1,1).

Quand aucune instanciation de variables ne satisfait toutes les clauses de la formule simultanément, nous disons que la donnée est contradictoire. Dans cette situation on doit chercher à déterminer une assignation aux variables de telle sorte qu'elle puisse satisfaire le plus grand nombre possible de clauses. Ce problème est connu comme le problème de satisfiabilité maximal ou MAX-SAT.

Quand des poids sont associés aux clauses, le but revient à maximiser la somme des poids de clauses satisfaites. Ce problème est nommé problème de satisfiabilité pondéré maximal ou MAX-W-SAT.

VI. La recherche locale

La recherche locale est une technique très utilisée en pratique pour aborder des problèmes d'optimisation combinatoire difficiles est d'améliorer de façon itérative une solution existante en explorant un voisinage de celle-ci. Les solutions dites voisines sont généralement obtenues en appliquant des transformations (aussi appelées mouvements) plus ou moins importantes (c'est-à-dire plus ou moins locales) à la solution courante.

Après un premier essor dans les années 70 lié aux travaux de *Lin et Kernighan* sur le problème du voyageur de commerce, la recherche locale a connu un regain d'intérêt à la fin des années 90 avec de nombreuses études expérimentales et quelques tentatives d'analyse théorique.

Les différents algorithmes de recherche locale se distinguent par la manière de choisir une solution dans le voisinage V(S) de la solution (c'est un ensemble de solution générée à partir de la modification d'une composante de S) de la solution courante S. **[DAP08]**

Parmi ces algorithmes, on peut citer : la méthode de descente, recherche tabou, recuit simulé et la méthode de voisinage (mouvement d'attraction) celle qui nous intéresse.

- ***Méthode de descente***

La méthode descente choisit a chaque étape la meilleure solution voisine, et elle s'arête des qu'elle ne peut pas améliorer la solution courante ou le nombre d'itération fixé est atteint. **[DAP08]**

- ***Le recuit simulé***

La méthode de recuit simulé s'inspire du recuit physique, le principe de cette méthode consiste a accepté d'aller vers une solution voisine S1 moine bonne que la solution courante s avec une certaine probabilité P(S,S1,T) ;cette probabilité dépond d'une part de l'écart de qualité entre S1 et s(plus S1 est de qualité voisine a s, plus elle a la chance d'être acceptée),et d'autre part du temps de calcul déjà effectuer(représenter avec l'intermédiaire du paramètre température T qui décroit avec le temps) . **[DAP08]**

- ***La Recherche tabou***

La méthode tabou a été développée par Glover.son principe et de choisir a chaque itération une solution Si+1 ϵ V(Si) .Quand un mouvement est effectué pour aller de Si a Si+1,On introduit le mouvement inverse dans une liste tabou LT de taille finie et il est interdit de faire ce mouvement(sauf si ce mouvement permet d'atteindre une solution de qualité supérieure a celle de la meilleure trouvée jusqu'alors) durant t itération(t donnée par l'utilisateur). **[DAP08]**

- ***Méthode de voisinage*** **[JKH99]**

Les méthodes de voisinage sont fondées sur la notion de voisinage. Nous allons donc introduire d'abord cette notion fondamentale ainsi que quelques notions associées.

Le voisinage d'une solution est un sous-ensemble de solutions qu'il est possible d'atteindre par une série de transformations données. Par extension on désigne parfois par le terme « *Voisinage* » l'ensemble des transformations considérées.

Une méthode typique de voisinage débute avec une configuration initiale, et réalise ensuite un processus itératif qui consiste à remplacer la configuration courante par l'un de ses voisins en tenant compte de la fonction de coût. Ce processus s'arrête et retourne la meilleure configuration trouvée quand la condition d'arrêt est réalisée. Cette condition d'arrêt concerne généralement une limite pour le nombre d'itérations ou un objectif à réaliser. Un des avantages des méthodes de voisinage réside précisément dans la possibilité de contrôler le temps de calcul : la qualité de la solution trouvée tend à s'améliorer progressivement au cours du temps et l'utilisateur est libre d'arrêter l'exécution au moment qu'il aura choisi. Les méthodes de voisinage diffèrent essentiellement entre elles par le voisinage utilisé et la stratégie de parcours de ce voisinage.

VII. Méthodes hybrides

Il arrive fréquemment que des solutions hybrides soient apportées au problème d'exploration de l'espace. L'idée est de combiner deux méthodes

complémentaires : une de recherche globale et une de recherche locale. En effet, les effets combinés obtenus par ce genre d'hybridation sont souvent encourageants.

Le mode d'hybridation qui semble le plus fécond concerne la combinaison entre les méthodes de voisinage et l'approche d'évolution. L'idée essentielle de cette hybridation consiste à exploiter pleinement la puissance de recherche de méthodes de voisinage et de recombinaison des algorithmes évolutifs sur une population de solutions Un tel algorithme utilise une ou plusieurs méthodes de voisinage sur les individus de la population pendant un certain nombre d'itérations ou jusqu'à la découverte d'un ensemble d'optima locaux et invoque ensuite un mécanisme de recombinaison pour créer de nouveaux individus.

VIII. L'hybridation ACO / Recherche locale

Dans la plupart des problèmes NP-Complets, les algorithmes de colonies de fourmis sont souvent plus efficaces quand ils sont *hybridés* avec des algorithmes de recherche locale. Ceux-ci améliorent la meilleure solution trouvée par les fourmis. Cette solution sera utilisée dans la mise à jour globale.

Du point de vue de la recherche locale, utiliser des algorithmes de colonies de fourmis pour générer 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. **[ACO08]**

IX. Conclusion

Dans ce chapitre nous avons présenté les problèmes combinatoires ainsi que quelque méthodes de résolution.les méthodes exactes sont très couteuses en terme de temps de calcule, elles deviennent inefficaces plus en plus que la taille des problèmes s'accroit et pour palier ceci en fait appelle aux méthodes approchées.

Nous avons aussi présenté en détaille La méthode choisi pour résoudre notre problème la métaheuristique de colonies de fourmi, son origine, ses application, et

son évolutions, et son efficacité lorsque elle est hybridé avec une recherche locale, Elle c'est montrée très efficace face à plusieurs problèmes NP-complet.

CHAPITRE 3

CONCEPTION

CHAPITRE3
CONCEPTION

1. Introduction

Le problème du repliement de protéine est un problème combinatoire extrêmement difficile, il fait ces dernières années l'objet de recherche très approfondies. C'est l'un des problèmes NP-difficile très célèbres.

Le paradoxe de Levinthal désespère toute tentative de résolution par énumération. Seules les méthodes approchées peuvent être employées à cette fin.

Dans notre travail nous allons adapter la métaeuristique des colonies de fourmis hybridé avec une recherche locale pour le problème de repliement de protéine et pour cela on essaiera de donner une formulation adéquate au problème ainsi qu'une modélisation plus au moins manipulable.

L'idée générale de notre travail à été inspirée des travaux de l'équipe OPAL **[ANG05].**

2. Formulations du modèle HP de DILL

Il existe deux méthodes pour déterminer la conformation native d'une protéine : la première (et la meilleure) serait de "photographier" aux rayons X la protéine dans une solution d'eau pure. L'inconvénient de cette méthode vient du coût de la technique.

La seconde technique, celle qui nous intéresse, est de modéliser ce problème comme un problème d'optimisation combinatoire et de le résoudre de manière exacte ou approchée en utilisant des algorithmes appropriés **[ANG05].**

Le problème étant très complexe, de nombreux modèles simplifiés existent, le

modèle choisi est le modèle HP proposé par Dill **[DIL95].**

Ce modèle possède les caractéristiques suivantes :

1. Les acides aminés (monomères) ont tous la même taille
2. Les acides aminés sont simplifiés. Il n'y en a pas plus que deux types : *les monomères hydrophobes*, notés H par la suite, et *les monomères hydrophiles ou polaires*, notés P par la suite.
3. Les liens entre deux monomères ont tous la même longueur.
4. Chaque monomère ne peut occuper une position que sur une grille en deux ou trois dimensions.
5. La fonction calculant l'énergie d'une protéine dans une conformation donnée, est décrite ainsi :

Soient a et b deux monomères de type H non consécutifs de la protéine et f la fonction telle que :

$$f(a, b) = \begin{cases} -1 & \text{si a et b sont voisins dans la conformation.} \\ 0 & \text{sinon.} \end{cases}$$

- L'énergie de la protéine dans cette conformation est égale à la somme des f (a, b) sur tous les couples a et b possibles de monomères H non consécutifs de la protéine.

Le problème consiste donc à chercher une conformation de moindre énergie. Ce qui revient à maximiser le nombre de contacts H-H.

La (**figure 3.1**) montre deux conformations possibles pour la séquence PPHPPHPPHH. La conformation de moindre énergie est celle qui maximise le nombre de contacts H-H.

A noter que, tout au long de ce chapitre, les monomères H seront représentés par un disque foncé et les monomères P par un disque blanc.

Pour pouvoir réaliser un mouvement « mouvement d'attraction », il y a deux cas possibles :

- P_C est libre **(figure 3.4*(a)*)**
- $P_C = S_{i-1}$ **(figure 3.4*(b)*)**

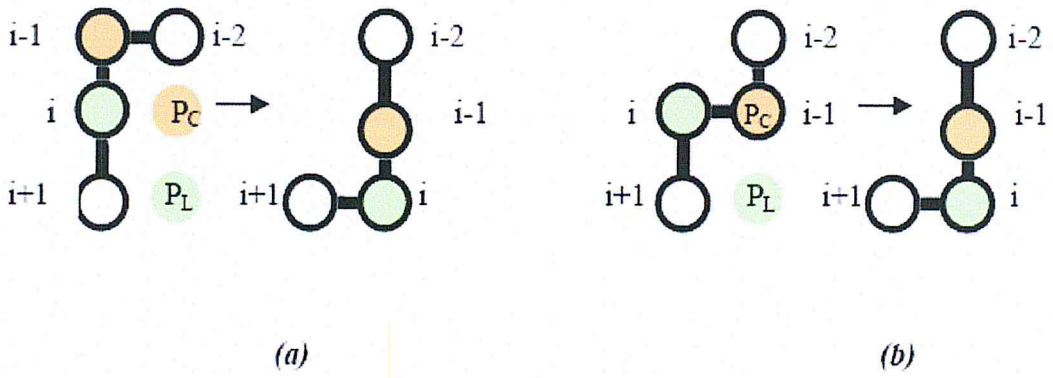


Fig. 3.4 Le mouvement « mouvement d'attraction »

On place alors S_i à la position P_L et on déplace les acides aminés S_{i-1}, S_{i-2} - - - jusqu'a atteindre une conformation. On peut bien sûr effectuer le mouvement dans l'autre sens.

Il y a potentiellement quatre mouvements « mouvement d'attraction » pour chaque acide aminé :

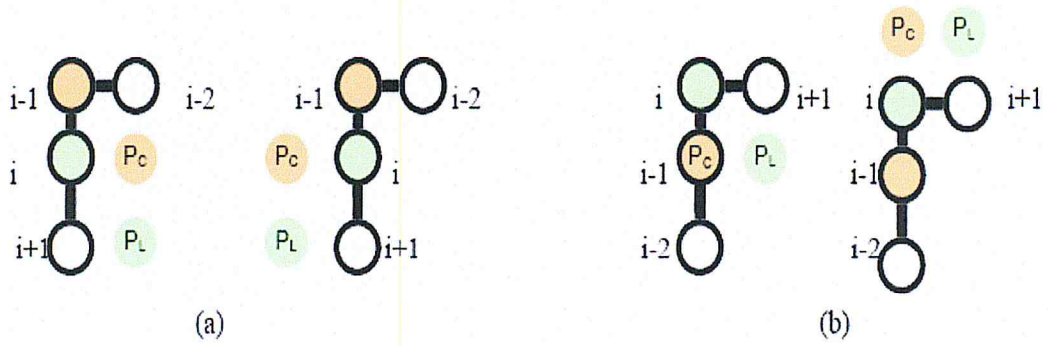


Fig. 3.5 Ensemble des mouvements « mouvement d'attraction » possibles pour deux monomères donnés S_i et S_{i-1}

En effet, ces quatre mouvements sont possibles, car on peut parcourir la séquence dans les deux sens (voir les cas (a) et (b) de la **(figure 3.5)**), et de plus les positions P_L et P_C peuvent être placées de chaque coté du monomère.

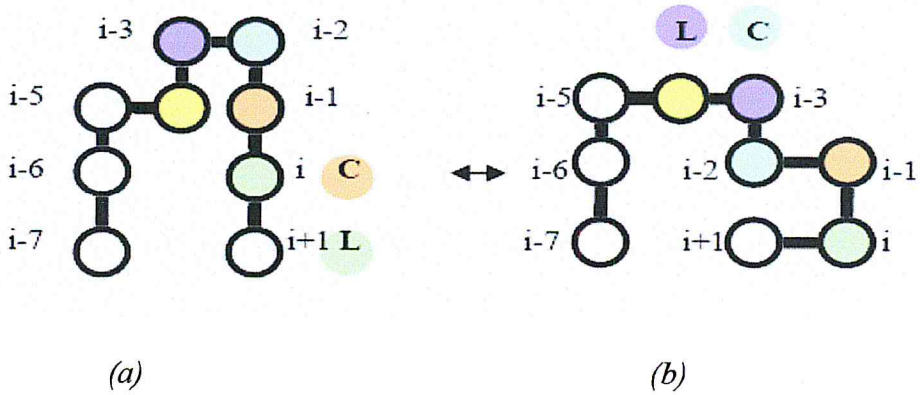


Fig. 3.6 Un mouvement « mouvement d'attraction » (Full- Move) sur un cas plus complexe

Dans la **(figure 3.6),** le mouvement « mouvement d'attraction » est plus complexe, car il implique un déplacement de quatre monomères de la séquence (le monomère jaune ne bouge pas). De plus, on peut remarquer que ce voisinage est réversible, comme le montre la (**figure 3.6)** La conformation *(a)* est obtenue en appliquant un mouvement « mouvement d'attraction » sur les monomères S_{i-3} et S_{i-2} a partir de la conformation *(b)*. Pour obtenir la conformation *(b)* a partir de *(a)*, on applique le mouvement « mouvement d'attraction » sur les monomères S_i et S_{i-1}.

4 .Approche de résolution du problème du repliement de protéines par les algorithmes de fourmis hybridé avec la recherche locale

De nombreuses méthodes de résolution qui abordent le problème du repliement de protéines sont basées sur le modèle HP de Dill. Parmi celles-ci, on peut citer des méthodes approchées comme les algorithmes génétiques, la méthode de Monte-Carlo, Monte-Carlo Evolutionnaire, la recherche taboue et *les colonies de fourmis* **[ANG05] .**

Nous allons adapter la métaheuristique des colonies de fourmis au problème du repliement de protéines

A chaque itération de l'algorithme, k fourmis construisent chacune une solution en prenant des décisions. Ces décisions sont prises à partir d'un critère heuristique, ainsi que sur les traces de phéromone. Ces traces sont mises à jour en fonction de la qualité des solutions obtenues. Elles sont renforcées pour les choix ayant donnés de meilleures solutions, et diminuées pour les autres.

En pratique, on répète les trois phases suivantes, jusqu'à atteindre un nombre maximum d'itérations fixé initialement :

- construction de k solutions initiales,
- application d'une recherche locale à chacune de ces solutions
- mise à jour du taux de phéromone en fonction des solutions trouvées **[ANG05].**

Shmygelska dans **[SHM 02] [SHM 05]** propose un algorithme de colonies de fourmis pour la résolution du problème de repliement de protéines. La différence entre cet algorithme et l'équipe OPAL réside dans la phase de construction et dans l'algorithme de recherche locale.

Dans la phase de construction de l'algorithme de Shmygelska **[SHM 02],** chaque fourmi commence le placement des monomères d'une conformation donnée a partir d'un monomère choisi aléatoirement alors que, dans le second, l'ordre d'apparition des monomères est respecté. Cette façon de procéder évite d'avoir à choisir le sens du parcours de la séquence initiale a partir du premier monomère placé. Autrement dit, on n'a pas à répondre a la question : faut-il placer d'abord les monomères qui se trouvent a gauche ou a droite du premier monomère choisi ?

Notre algorithme **HACORL** *(Hybridation de l'Algorithme de Colonies de fourmis avec la Recherche Locale) s'appuis sur cette deuxième stratégie avec l'hybridation d'une recherché locale.*

4.1 Principe de l'algorithme proposé

Les fourmis construisent des conformations possibles pour une protéine et actualisent les taux de phéromone à partir de la qualité des solutions obtenues. Les conformations d'une protéine sont modélisées par une série de chiffres qui représentent la direction relative d'un monomère par rapport aux deux monomères précédents le long de la séquence. Un « 0 » indique que le monomère est place à gauche par rapport aux deux monomères précédents (respectivement « 1 » pour la direction « avant » et « 2 » pour la direction « droite »).

Comme les conformations ne changent pas si on effectue une rotation, la position des deux premiers monomères peut être fixée sans problème.

Par exemple **(figure 3.7)**, une conformation possible de la séquence 1 **[DIL95]** (HPHPPHHPHPPHPHHPPHPH) peut être représentée par la séquence (0100220100202200 10).

On remarque que, pour une protéine de longueur n, la conformation associée a une longueur de (n-2) motifs.

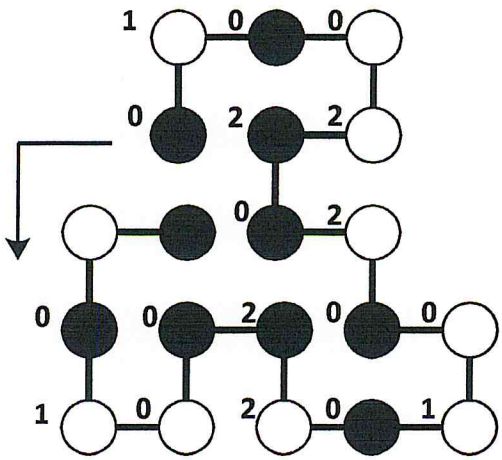


Fig3.7 Représentation d'une conformation d'une protéine a l'aide des coordonnées relatives

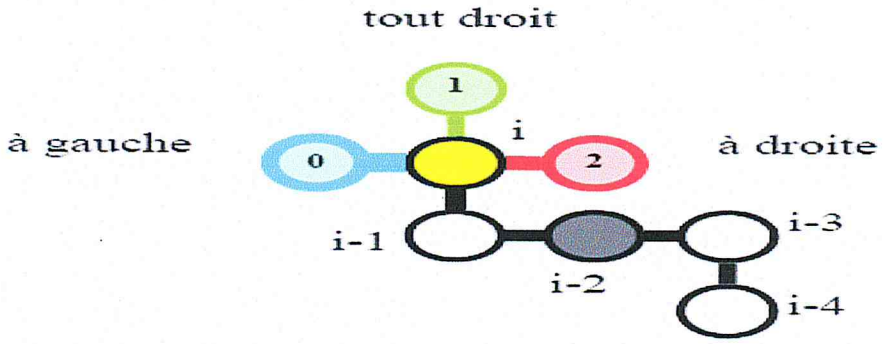


Fig. 3.8 Choix du placement d'un monomère en fonction du taux de phéromone

Pour chaque monomère S_i et pour chaque direction $d \in \{0, 1, 2\}$, on dispose d'un taux de phéromone $\tau_{i,d}$. Ce taux de phéromone intervient pour déterminer la probabilité qu'a une fourmi de placer le monomère S_{i+1} dans la direction d par rapport aux monomères S_i et S_{i-1} sachant que le monomère S_i est le dernier monomère place. Ainsi, dans la **(figure 3. 8)**, nous avons indiqué les choix possibles pour le prochain monomère S_{i+1} à placer. L'épaisseur des traits correspond au taux de phéromone. Plus la quantité est grande pour une direction, plus l'agent (ou fourmi) choisit cette direction. Ici, on à plus de chances de voir le prochain monomère place à la gauche du dernier monomère S_i placé.

4.2 L'organigramme du HACORL

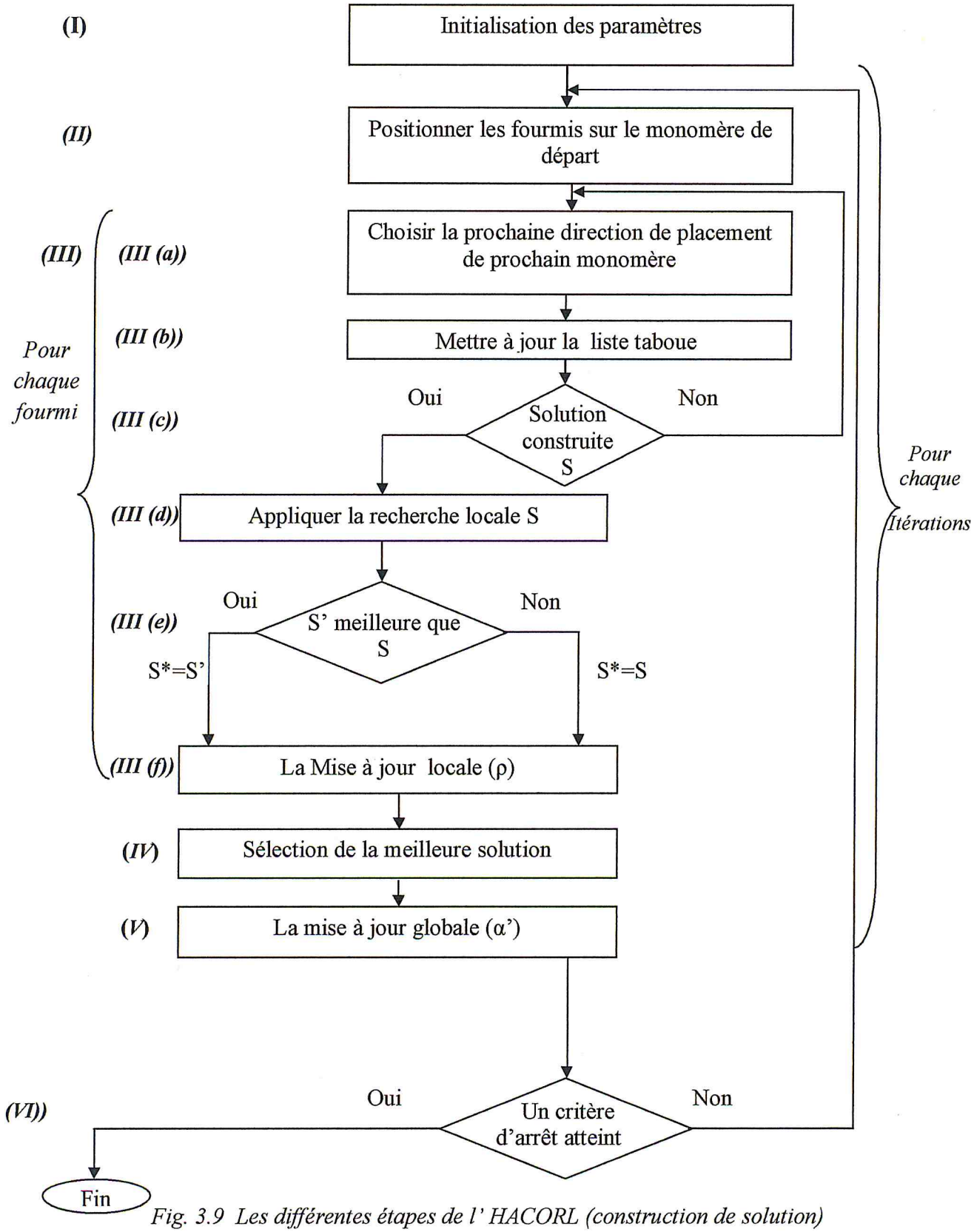


Fig. 3.9 Les différentes étapes de l' HACORL (construction de solution)

Explication de l'organigramme :

I. Initialisation des paramètres : spécification des paramètres (α, β, nombre d'itération et le nombre des fourmis),initialisation la valeur de phéromone qui égale a un taux ρ redéfini τ_0 .

II. Initialisation des fourmis : positionner les fourmis sur le monomère de départ.

III. Construction de la solution : chaque fourmi construit une conformation complète, en répétant (a, b, c, d, e) jusqu'à termin.la chaine de la protéine alors :

III. a le chois du composant : choisir la direction prochain monomère de la séquence en entrée, en appliquant la règle de probabilité.

III. b Mettre le composant dans la liste tabou : une fois la composant choisi, on le met dans la liste tabou pour sauvegarder le chemin choisi, et mise a jours la valeur de la phéromone (cette liste correspond a la mémoire interne de la fourmi, c .a. d permet pour chaque fourmi de sauvegarder la solution construite).

III. c La solution construite : si la fourmi n'a pas terminer la construction de la solution (elle n'a pas atteint la fin de la séquence) elle répète le processus de construction depuis l'étape III .a

III. d Appliquer la recherche locale sur la solution obtenue (S) par chaque fourmi.

III. e SI le résultat trouvée (S') est meilleur que la solution obtenue(S) alors retourner S' sinon garder (S).

III. f La mise à jours locale : consiste à déterminer le taux de phéromone τ0 sur le composant, pour assurer la notion de la diversification de la solution (une meilleure exploration de l'espace de recherche) c. a. d la prochaine fourmi

IV. A la fin de chaque itération sélectionner la meilleure solution parmi n solutions construites

V. Appliquer la mise à jour globale sur la solution trouvée

VI. SI le critère d'arrêt atteint (nombre d'itération atteint ou optimum atteint) alors retourné la meilleur solution.

4.3 L'algorithme HACORL

Algorithme: ***Colonie_Fourmis***
Début
Initialiser Paramètres
Initialiser meilleure solution à 0
Pour chaque itération **faire**
Pour chaque fourmi **faire**
Pour chaque monomère, **faire**
Calculer les paramètres et déterminer la probabilité
Pour chaque direction
Choisir une direction
Fait
Si la conformation est valide
Alors Effectuer une recherche locale (voisinage "mouvement d'attraction")
Comparer à la meilleure solution
Si la solution trouvée est meilleure
Alors elle devient la meilleure solution
Sinon rien
Fin si
Sinon rien
Fin si
Mise à jour locale
Fait
Réactualiser le taux de phéromone (mise à jour globale)
Fait
Retourner la meilleure solution

FIN

4.4 Eléments de l'algorithme de HACORL

4.4 .1 Eléments de l'algorithme de colonie de fourmis

Lors de la phase de construction de l'algorithme, les deux premières acides amines sont fixées au départ sur la grille. Chaque fourmi commence donc à partir de la troisième acide amine de la séquence de la protéine donnée. A partir de cette position, la fourmi décide de suivre une direction relative à chaque étape de construction de l'algorithme, déterminée suivant une loi de probabilité basée sur des taux de phéromone et des valeurs heuristiques.

- ***Paramètres heuristiques employés*** **[ANG05]**

Pour chaque monomère S_i et une direction $d \in \{0, 1, 2\}$, une valeur de la phéromone $\tau_{i,d}$ est définie. Néanmoins, il n'est pas rare de trouver dans les algorithmes de colonies de fourmis d'autres paramètres servant a guider les fourmis dans leur phase de construction d'une solution. Ces autres paramètres sont communément désignés sous le nom de « paramètres heuristiques ». Ainsi, nous utilisons une valeur heuristique $\mu_{i,d}$ dont le but est de guider la recherche vers de bonnes conformations. Cette valeur heuristique est basée sur $h_{i,d}$ le nombre de nouveaux contacts H-H obtenus par rapport aux monomères H déjà placés, en plaçant le monomère S_{i+1}, dans la direction d, relativement a S_i et S_{i-1}.

Nous définissons $\mu_{i,d} = h_{i,d} + 1$, pour éviter le problème de division par zéro.

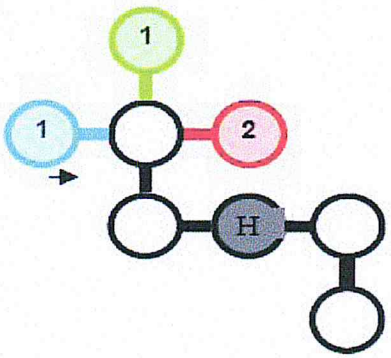


Fig3.10 Choix de placement d'un monomère, en fonction du nombre de contacts possibles.

Dans la **(figure *3.10)*,** la valeur de $h_{i,d}$ est inscrite a l'intérieur de chaque cercle et on suppose que le monomère à placer est un H. Si on place le monomère à droite, il y aura un contact, d'ou la valeur de 2.

- ***Calcul des probabilités***

Pendant la phase de construction, pour étendre une conformation partielle $S1 \ldots Si$ a $Si+1$, une fourmi choisit de suivre la direction relative $d \in \{0, 1, 2\}$ d'après la probabilité suivante:

$$p_{i,d} = \frac{\tau_{i,d}^{\alpha} * \mu_{i,d}^{\beta}}{\sum_{e \in \{0,1,2\}} \tau_{i,e}^{\alpha} * \mu_{i,e}^{\beta}}$$

$\tau_{i,d}$: *valeur de phéromone*

$\mu_{i,d}$: *valeur heuristique*

α et β : déterminent l'influence relative du taux de phéromone et du paramètre heuristique.

- ***Les pistes de phéromones***

On a déjà vu que l'utilisation des pistes de phéromone est capitale pour un algorithme de colonies de fourmis.

Pour notre algorithme HACOLR, la phéromone est déposée sur la direction choisi qui relie deux monomères. La piste de phéromone $\tau_{i,d}$ entre les deux monomères Si et $Si+1$ permet a une fourmis d'avoir une historique sur les différentes directions choisis par ses congénères, vers le prochain monomère, et cela a partir du monomère courant.

L'information phéromone est implémentée par une matrice (n-2)* 3. Cette matrice constitue une mémoire collective de toutes les fourmis de la colonie. Toutes les entrées de la matrice phéromone sont initialisées à une petite valeur τ_0 généralement égale à 0,1.

La mise à jour de la phéromone se fait à deux niveaux dans l'algorithme :

- Premièrement, durant la construction de solution et plus exactement après l'ajout

d'un monomère à la solution partielle les fourmis « mangent » une quantité de phéromone des connexions visitées **[HAM03]**, c.à.d. il s'agit de diminuer la quantité de phéromone sur l'arc reliant les deux monomères si la fourmi a choisi de passer de S_i à S_{i+1} en prenant la direction d. Ce comportement est appelé *locale phéromone update (l'évaporation)*. Cette mise à jour se fait selon la formule suivante:

$$\tau_{i,d} = \tau_{i,d} * (1 - \rho)$$

Ce processus assure l'exploration d'autres solutions par les fourmis successeurs.

- Deuxièmement, la mise à jour globale qui se fait à la fin de chaque construction des solutions par les fourmis, sur le chemin de la meilleure fourmi (celle ayant trouvé la meilleure conformation).On actualise le taux de phéromone à partir de la qualité de cette solution, de la manière suivante :

$$\tau_{i,d}^{+} = \tau_{i,d} * (1-\alpha) + \alpha * \Delta\ taux$$

Où :

$1 \in [0,1]$: la persistance des pistes de phéromones.

$\alpha' \in [0,1]$: le paramètre fixant la décadence de la phéromone.

$$\Delta\ taux = \begin{cases} E_s / E^* & \text{si l'optimum est connu} \\ & (E^* \textit{ est l'optimum selon les benchmarks de la littérature}) \\ |E_s| * 0.01 & \text{si l'optimum n'est pas connu} \end{cases}$$

Ces deux stratégies de mise à jour globale s'expliquent par le fait d'appliquer l' HACORL à deux types de séquences différents. En effet, notre algorithme a été conçu au début pour résoudre des instances figurants dans les benchmarks de la littérature. Donc l'énergie optimale est connue au préalable. Comme il sera illustré dans le chapitre des tests, l' HACORL a donné des résultats très satisfaisant pour ce genre de séquences.

On a tenté par la suite de prolonger notre algorithme à des séquences arbitraires (saisies par l'utilisateur) où l'énergie optimale n'est pas connue, ce qui explique la modification dans la formule de mise à jour offline.

4.4 .2 Recherche locale

Dans notre approche la solution initiale de la recherche locale est une solution trouvée par une fourmi.

Trouver une solution initiale (trouver par une fourmi)

Trouver une solution au voisinage de la solution initiale (en appliquant mouvement d'attraction)

Si (la solution trouvée est meilleure que l'initiale)

Alors remplacer la solution initiale.

---

5. Conclusion

Dans ce chapitre nous avons adapté la métaheuristique de colonie de fourmi pour la réalisation de problème de repliement de protéine, cet algorithme est hybridé avec une recherche locale afin d'amélioré la solution trouvé. Les éléments essentiels de cette adaptation sont : le choix du modèle de la protéine, le choix des structures des données pour la représentation de la solution.

Notre application a été implémenté en JAVA, le chapitre suivant sera consacré a la présentation du logiciel, aux choix des paramètres et aux testes sur certaine séquence de benchmarks.

CHAPITRE 4

ETUDE EXPÉRIMENTALE

CHAPITRE4

ETUDE EXPÉRIMENTALE

1. Introduction

Ce chapitre est consacré à la présentation de l'application de notre travail Cette dernière a été développée en *Java* sous l'environnement *NetBeans* version *6.9,* Elle permet de prédire la structure native d'une protéine à partir de sa séquence d'acides aminés par la métaheuristique des colonies de fourmis hybridée avec une recherche locale.

Nous commençons par la présentation de l'interface utilisateur et ces différentes fonctionnalités.

Nous procédons par la suite aux différents jeux de tests effectués. Tous les tests ont été réalisés sur core2 Duo processor 2.00GHz, avec 3 Go de RAM.

2. Présentation du logiciel

2.1 Présentation du langage Java

Java est un langage de programmation développé par Sun Microsystems, il est très utilisé, notamment par un grand nombre de programmeurs professionnels, ce qui en fait un langage incontournable actuellement.

Une de ses plus grandes forces est son excellente portabilité : une fois le programme est créé, il fonctionnera automatiquement sous Windows, Mac, Linux ... etc. En fait, les programmes Java ne savent même pas où ils s'exécutent, car ils le font à l'intérieur d'une enveloppe logicielle spéciale appelée Machine Virtuelle Java (JVM). **[BA09]**

2.2 Présentation de l'environnement NetBeans

NetBeans est un environnement de développement intégré ***(EDI)*** pour Java, placé en open source par Sun en juin 2000 En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML,,

PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage , éditeur graphique d'interfaces et de pages Web). L'EDI s'exécute sous diverses plates-formes, y compris Windows et Linux. Il est simple à installer et à utiliser. Il offre aux développeurs tous les outils nécessaires à la création d'applications de bureau, d'entreprise, Web et mobiles professionnelles pouvant être utilisées sur diverses plates-formes.

2.3 Description du logiciel

2.3.1 L'interface utilisateur

L'interface utilisateur de notre logiciel est une interface graphique conviviale et très facile à utiliser. Elle permet d'accéder aux différentes fonctionnalités du logiciel.

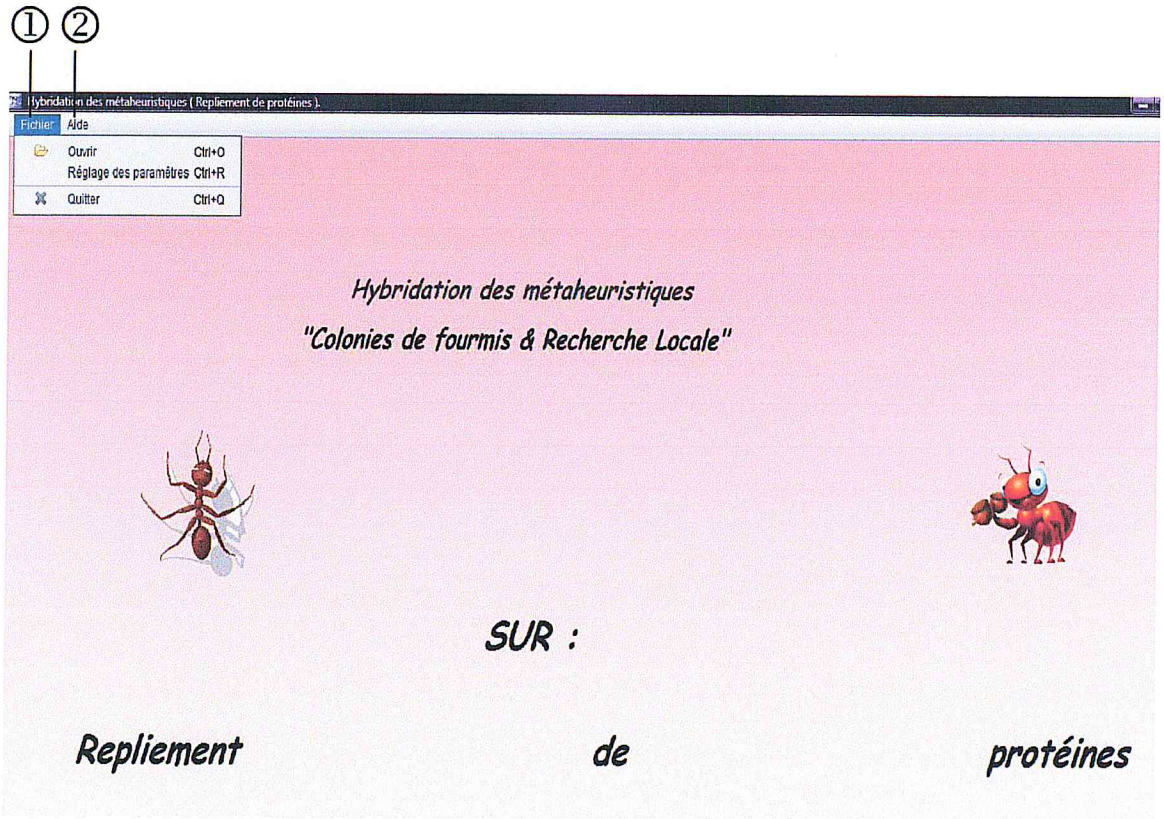


Fig.4.1 Interface utilisateur du logiciel

Notre interface utilisateur comporte un composant principal : une barre de menus.

Le tableau suivant décrive des différentes fonctionnalités de la barre de menu

N°	Menu	Sous- menu	Explications
①	**Fichier**	Ouvrir	*Afficher la liste des protéines enregistrées par l'utilisateur (accès a la base de données)*
		Réglage de Paramètres	*Afficher une fenêtre qui affiche à l'utilisateur les paramètres d'exécution de l'algorithme, et lui permet de modifier les paramètres et de lancer l'exécution.*
		Quitter	*Quitter l'application.*
②	**Aide**	A propos	*A propos de l'application.*

Tableau 4.1 Description des différentes fonctionnalités de la barre de men

2.3.2 Réglage de paramètres

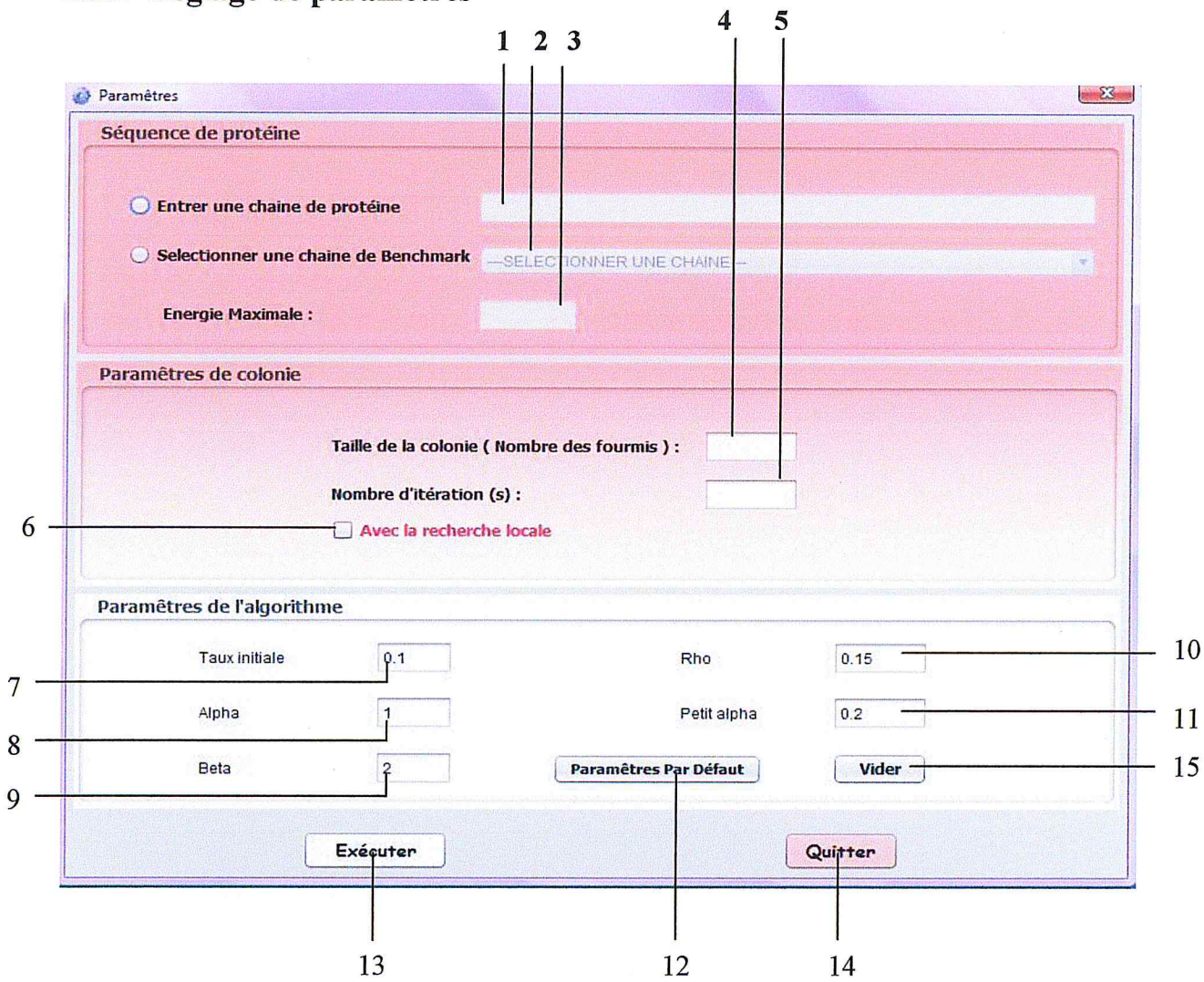


Fig4.2 Fenêtre de réglage de paramètres

1 ⟶ Zone de saisie d'une séquence de protéine.

2 ⟶ Choix d'une séquence des benchmarks

3 ⟶ L'énergie de la séquence introduite ou la séquence benchmarks

4 ⟶ Tailles de la colonie ou nombre de fourmis

5 ⟶ Nombre d'itération

6 ⟶ Zone à coché pour l'hybridation avec la recherche locale

7 ⟶ Valeur de τ_0, le taux de phéromone initial

8 ⟶ Valeur de α, le paramètre qui définie l'influence du taux de phéromone.

9 ⟶ Valeur de β, le paramètre qui définie l'influence de l'information heuristique

10 ⟶ Valeur de ρ, le paramètre fixant le taux d'évaporation de la phéromone (mise à jour locale)

11 ⟶ Valeur de α', le paramètre fixant le taux de décadence de la phéromone (mise à jours globale)

12 ⟶ Choisir les paramètres par défauts prédéfinis dans le programme

13 ⟶ Lancer l'exécution de l'algorithme avec les paramètres affichés

14 ⟶ Fermer la fenêtre de réglage des paramètres

15 ⟶ vider les cases des paramètres

2.3.3 Solution

Lorsqu'on clique sur le bouton *Exécuter*, les détails des solutions trouvées apparaissent alors dans une fenêtre indépendante :

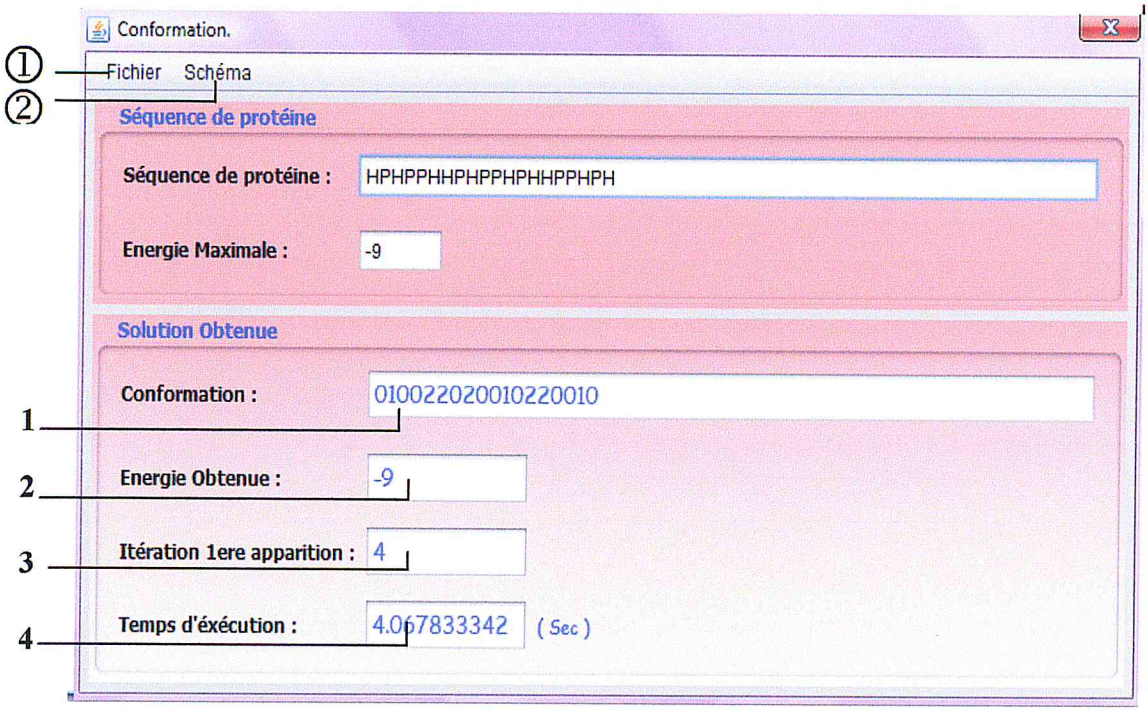


Fig4. 3 Fenêtre de solution

1 → La séquence qui représente la conformation (structure native) correspondante à la solution

2 → L'énergie de la conformation trouvée

3 → L'itération de la première apparition de l'énergie optimale

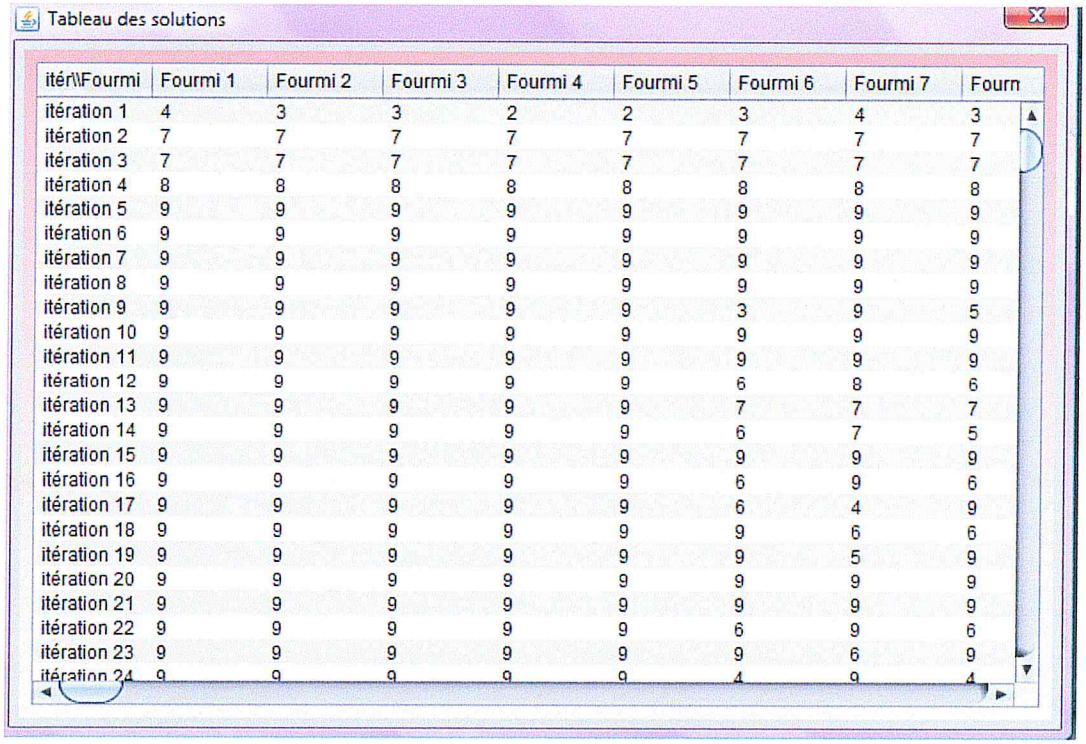
4 → Le temps nécessaire pour trouver cette solution

Les fonctionnalités de la barre de menus dans la fenêtre solution

① → Fichier → Enregistrer (enregistrer la solution dans la base de données /Quitter (Quitter la fenêtre solution)

② → Schéma → Tableau des solutions /Graphe des solutions /Dessiner la conformation

2.3.4 Tableau des solutions



itér\\Fourmi	Fourmi 1	Fourmi 2	Fourmi 3	Fourmi 4	Fourmi 5	Fourmi 6	Fourmi 7	Fourn
itération 1	4	3	3	2	2	3	4	3
itération 2	7	7	7	7	7	7	7	7
itération 3	7	7	7	7	7	7	7	7
itération 4	8	8	8	8	8	8	8	8
itération 5	9	9	9	9	9	9	9	9
itération 6	9	9	9	9	9	9	9	9
itération 7	9	9	9	9	9	9	9	9
itération 8	9	9	9	9	9	9	9	9
itération 9	9	9	9	9	9	3	9	5
itération 10	9	9	9	9	9	9	9	9
itération 11	9	9	9	9	9	9	9	9
itération 12	9	9	9	9	9	6	8	6
itération 13	9	9	9	9	9	7	7	7
itération 14	9	9	9	9	9	6	7	5
itération 15	9	9	9	9	9	9	9	9
itération 16	9	9	9	9	9	6	9	6
itération 17	9	9	9	9	9	6	4	9
itération 18	9	9	9	9	9	9	6	6
itération 19	9	9	9	9	9	9	5	9
itération 20	9	9	9	9	9	9	9	9
itération 21	9	9	9	9	9	9	9	9
itération 22	9	9	9	9	9	6	9	6
itération 23	9	9	9	9	9	9	6	9
itération 24	9	9	9	9	9	4	9	4

Fig4.4 Tableau des solutions trouvées par chaque fourmi dans chaque itération

2.3.5 Graphe des solutions

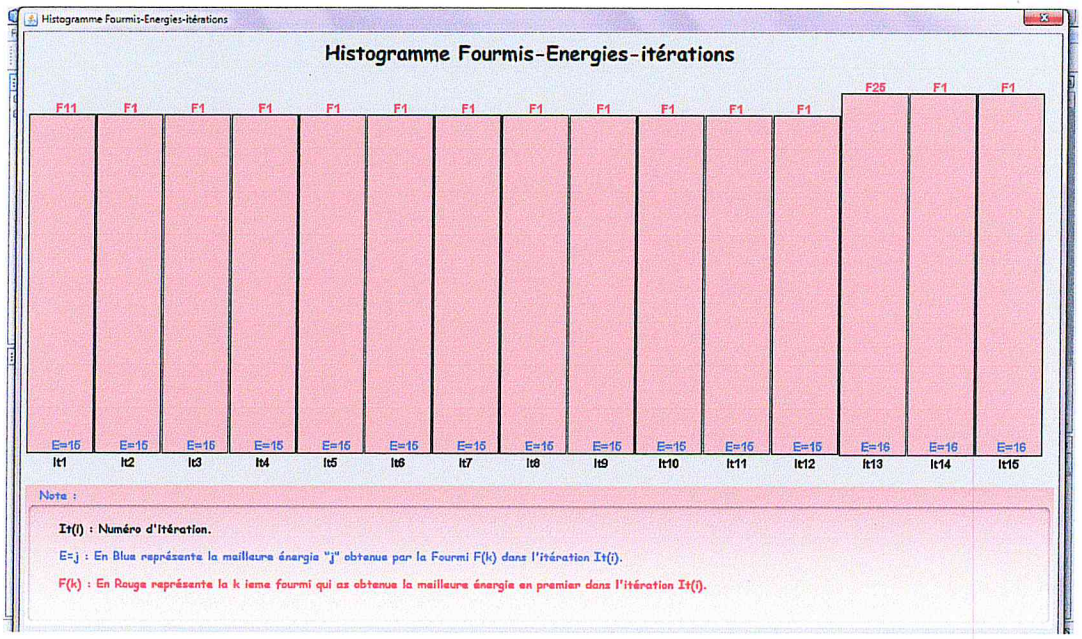


Fig4.5 Graphe des meilleures solutions trouvées dans chaque itération

2.3.6 Dessin d'une conformation

Lorsqu'on clique sur dessiner la conformation, le dessin de la conformation apparait alors dans une fenêtre indépendante :

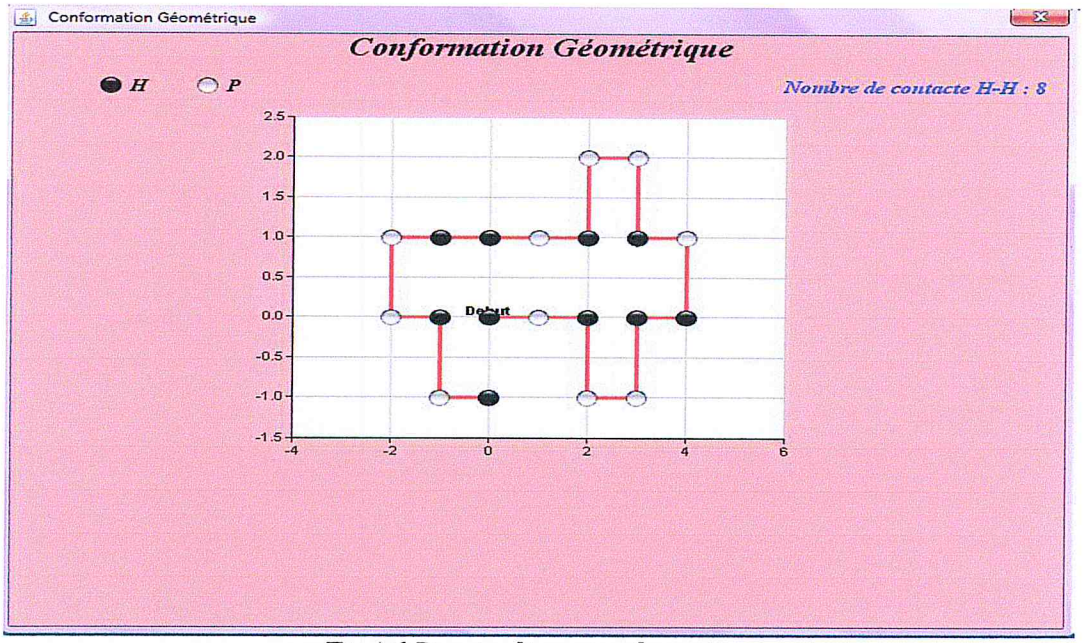


Fig4.6 Dessin d'une conformation

3. Tests et résultats

Pour analyser les performances de l'algorithme développé en Java sous l'environnement NetBeans, une série de tests ont été effectués sur des séquences de Benchmarks de la littérature. Ces tests ont été réalisés sur :

- Intel core2-duo processor T64002 ,2GHz, et 3 Go de RAM fonctionnant sous Windows Vista.
- Intel core2-duo processor T5670 ,1.8 GHz, et 3 Go de RAM fonctionnant sous Windows 7.
- Intel core2-duo processor T3400 ,2.16GHz, et 2 Go de RAM fonctionnant sous Windows Vista.
- Intel CPU N 270 ,1.60GHz, et 1Go de RAM fonctionnant sous Windows XP.

3.1 Résultat de la littérature

Le tableau suivant présente les benchmarks **[AB09]** qui nous ont servi lors des différentes expérimentations de notre algorithme. Ce tableau présente les numéros de séquences *(N°séq)*, les chaines de protéines, la longueur *(Long)* .ainsi que la meilleure solution connue (NB_contacts), pour chaque séquence, en termes de nombre de contacts H-H.

Une comparaison aux résultats des benchmarks est nécessaire pour le réglage des paramètres.

N° Séq	Protéine	Long	Energie optimale
01	$(HP)_2PH(HP)_2(PH)_2HP(PH)_2$	20	-9
02	$H(HP_2)_7H_2$	24	-9
03	$(P_2H)_2H(P_4H_2)_3$	25	-8
04	$P(P_2H_2)_2P_5H_7P_2H_2P_4H(HP_2)_2$	36	-14
05	$P_2H(P_2H_2)_2P_5H_{10}P_5(H_2P_2)_2HP_2H_5$	48	-23
06	$H_2(PH)_4H_3PH(P_3H)_2P_4(HP_3)_2HPH_4(PH)_4H$	50	-21
07	$P_2H_3PH_8P_3H_{10}PHP_3H_{12}P_4H_6PH_2PHP$	60	-36
08	$H_{12}(PH)_2((P_2H_2)_2P_2H)_3(PH)_2H_{11}$	64	-42
09	$H_4P_4H_{12}P_6(H_{12}P_3)_3HP_2(H_2P_2)_2HPH$	85	-53
10	$P_3H_2P_2H_4P_2H_3(PH_2)_3H_2P_8H_6P_2H_6P_9HPH_2PH_{11}P_2H_3PH_2HPP_2HPH_3P_6H_3$	100	-50
11	$P_6HPH_2P_5H_3PH_5PH_2(P_2H_2)_2PH_5PH_{10}PH_2PH_7P_{11}H_7P_2HPH_3P_6HPHP_2$	100	-48

Tableau4 .2 Benchmarks de la littérature **[AB09]**

3.2 Réglages de paramètres

Notre algorithme possède beaucoup de paramètres, chaque paramètre peux prendre de nombreuse valeurs différentes .il à été alors nécessaire de faire quelques choix affin de procéder à nos tests.

Pour fixé les valeurs des paramètres de notre algorithme, nous avons été guidés par les testes effectuer par la littérature.

- Le paramètre α : Indicateur de l'importance accordée aux phéromones locales lors de la prise de décision d'une fourmi, dans la littérature sera généralement fixé à 1.
- Le paramètre β : Indicateur de l'importance de la valeur heuristique, ce paramètre prendra soit 2 soit 5.
- Le paramètre ρ : Facteur d'évaporation de phéromone, dont la valeur sera de 0.15.
- Le paramètre α' : un paramètre qui fixe la décadence de phéromone.

Le taux de phéromone initiale est fixé à $\tau 0$.

Le nombre maximal des itérations pour chaque test est égal à 1000

Le **tableau 4 .3** résume les différents résultats obtenus avec différentes approches de résolution proposées dans la littérature. L'environnement matériel utilisé n'est pas connu pour la méthode de Monte-Carlo **[ACO08]**, mais toutes les autres ont été exécutées sur un Intel Pentium III cadencé à 1GHz, sauf pour la méthode Gtabu **[ACO08]**, qui a été utilisée sur un processeur Alpha à 1GHz et ACO **[ACO08]**, qui a été exécutée sur un Pentium 4-2.4GHz, 256Kb de mémoire cache et 1Mb de RAM.

On remarque que l'ensemble de ces méthodes marche très bien pour les protéines de taille les allant jusqu'à 50 acides aminés (séquences de 1 à 6). Au-delà, il existe une divergence entre les méthodes. Seuls les articles les plus récents

[SHM05] utilisent des séquences de taille plus grande (la puissance des machines aide).

Les colonies de fourmis semblent donner de bons résultats aussi.

N°Séq	Lg	Nb_contact	MC [UNG 93]	EMC [LIA01]	AG [UNG93]	ACO [SHM03]	ACO [SHM05]	PERM [HSU03]	AG+ Tabou [JIA 03]	GTabu [LES 03]
1	20	9	9	9	9	9	9	9	9	NT
2	24	9	9	9	9	9	9	9	9	NT
3	25	8	8	8	8	8	8	8	8	NT
4	36	14	13	14	12	14	14	14	14	NT
5	48	23	20	23	22	23	23	23	23	NT
6	50	21	21	21	21	21	21	21	21	NT
7	60	36	33	35	34	36	36	36	35	NT
8	64	42	35	39	37	42	42	42	39	42
9	85	53	NT	NT	NT	51	53	53	NT	53
10	100	50	NT	NT	NT	47	49	50	NT	50
11	100	48	NT	NT	NT	47	47	48	NT	48

Lg : longueur de la protéine ; Mc : Monte_Carlo ; EMC : Monte_Carlo Evolutionnaire ; AG : Algorithme Génétique ; ACO : Colonies de Fourmis ; PREM : Pruned enriched Rosenbluth method ; Tabou : méthode Tabou. NT : Non Testé

Tableau 4.3 Quelques résultats de la littérature **[ANG05]**

3.3 Influence de l'information heuristique et valeur de phéromone

Pour géré l'intensification, il faut passer par le réglage de paramètre β, et Pour géré la diversification, il faut passer par le réglage de paramètre α. Pour déterminer l'influence de l'information heuristique et la valeur de phéromone sur

notre algorithme, une série de tests à été effectué sur la quatrième séquence de benchmarks.

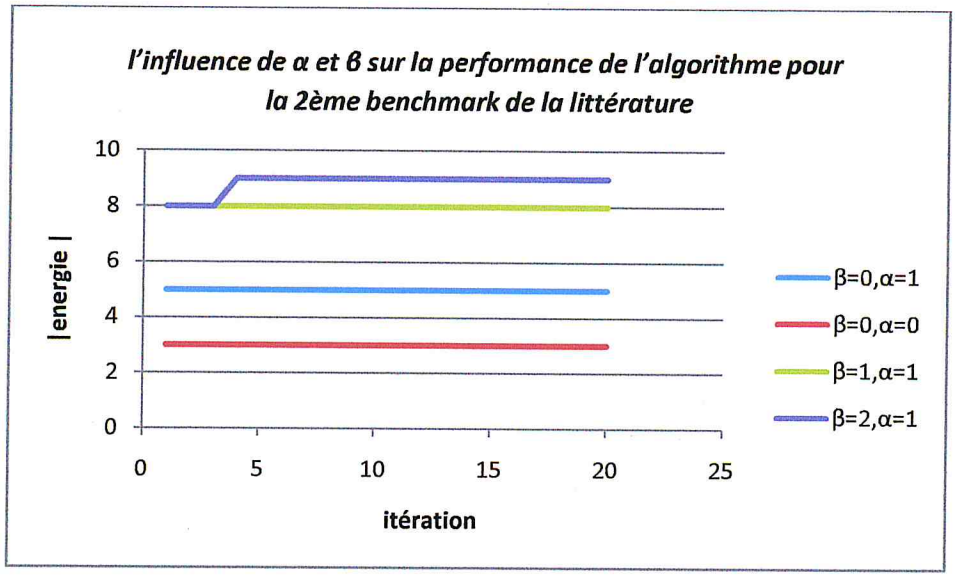


Fig4.7 l'influence de α et β sur la performance de l'algorithme pour la 2ème benchmark de la littérature

3.4 Résultats de l'exécution de l'algorithme HACORL sur des benchmarks

a. *Application de l'algorithme sur des séquences de benchmarks*

Notre algorithme a été testé sur des séquences de benchmarks (voir **Tableau4.2**), l'énergie de ces séquences est connue, ce qui nous a permit d'appliquer la formule de mis a jours globale du taux de phéromone suivante

$$\forall i,d \quad \tau_{i,d}{}^{+} = \tau_{i,d} * (1-\alpha') + \alpha' * \Delta\, taux$$

Où :

$\rho \in [0,1]$: la persistance des pistes de phéromones.

$\alpha' \in [0,1]$: le paramètre fixant la décadence de la phéromone.

$\Delta\ taux = E_s / E$ (E^* *est l'optimum de benchmark)*

Le tableau suivant résume les résultats obtenus par l'algorithme des fourmis sans et avec l'hybridation de la RL avec (α=1),(β=2) ,(ρ=0.15),(α'=0.2) et pour la 3éme séquence (α'=0.002).

N° seq	Lg	Energie Optimale	résultat obtenue SANS RL				résultat obtenue AVEC RL			
			Energie obtenue	It de 1ère APP	Nbr de fourmis	Temps d'exécu-tion	Energie obtenue	It de 1ère APP	Nbr de fourmis	Tempsd'exécu-tion
1	20	-9	**-9**	134	120	2.776	**-9**	380	120	9.3127
2	24	-9	**-9**	3	42	1.118	**-9**	14	42	1.787
3	25	-8	**-7**	275	73	4.860	-8	173	73	7.583
4	36	-14	**-14**	11	61	4.231	**-14**	166	61	6.503
5	48	-23	-17	14	61	8.238	-22	125	61	10.873
6	50	-21	-16	186	66	20.632	-20	489	66	27.918
7	60	-36	-32	179	79	33.125	-34	147	79	44.518
8	64	-42	-36	79	74	83 ,17	-37	478	74	99 ,82
9	85	-53	-44	58	44	35.793	-48	946	44	46.524
10	100	-48	36	108	69	75.380	-41	597	69	94.958
11	100	-50	-39	96	37	41.25	-42	503	37	53.656

Tableau4 .4 Résultats obtenus par HACORL pour les séquences de benchmarks

On remarque que la recherche locale à amélioré plusieurs solution, et qu'on a pue obtenir les optima de plusieurs séquences.

b. Exemple de représentation de solution

L'optimum de la première séquence benchmark, l'optimum est atteint après 380 Itérations à 120 fourmis et après l'application de la recherche locale.

La séquence de l'énergie maximale est :

« 010022020010220010»

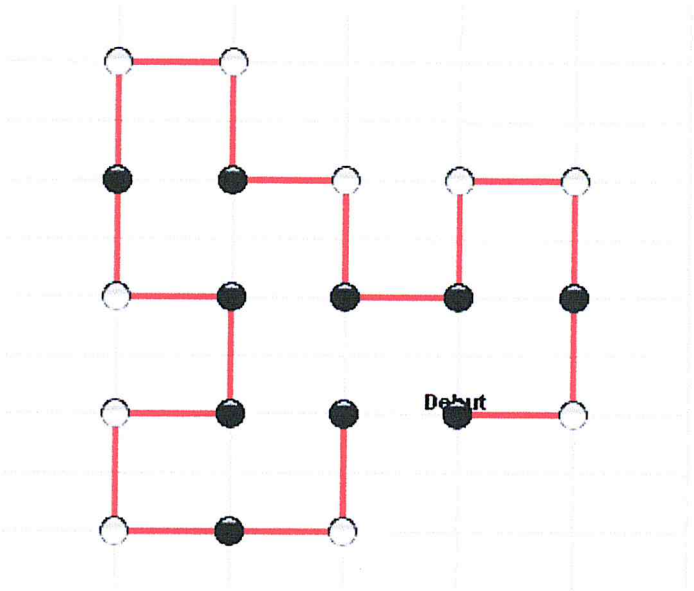


Fig4.8 Conformation trouvée pour la 1ère séquence de benchmarks

c. *Influence du nombre de fourmis*

Pour montrer l'influence du nombre des fourmis, nous avons utilisé une séries des testes tél que on fixe les paramètres α=1, β= 2 et ά= 0.2 et on varie le nombre des fourmis sur l'intervalle [94 ……102].

On exécute l'algorithme sur la 4éme séquence (voir tableau 4.5)

Nombre de fourmis	**Sans RL**			**Avec RL**		
	Energie obtenue	**It de 1ère APP**	**Temps d'exécu-tion**	**Energie obtenue**	**It de 1ère APP**	**Temps d'exécu-tion**
54	-9	1	10.705	-10	12	43.677
55	-10	20	9.163	-10	12	9.800
56	-9	1	4.239	-9	1	6.864
57	-9	1	4.29	-9	1	7.265
58	-9	1	4.32	-9	1	6.954
59	-10	4	4.395	-10	2	7.144

60	-10	13	4.55	-12	73	7.49
61	-11	27	4.641	**-14**	**166**	**7.26**
62	-14	237	4.66	-12	352	7.129

Tableau4 .5 L'influence nombre de fourmis sur qualité de solution pour la 4éme benchmark

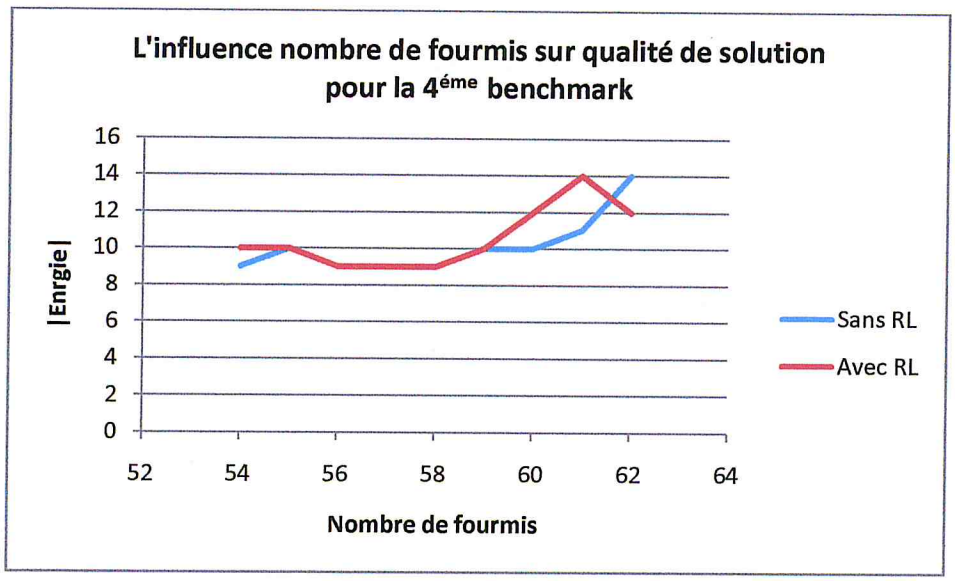


Fig4 .9 variation de nombre de fourmis pour la 4éme séquence benchmark

d. Application de l'algorithme sur des séquences aléatoire

Pour vérifier l'efficacité de l'algorithme pour ce genre de séquence, on la testé sur des séquences benchmarks, et on a comparé les résultats

La formule de mis à jours globale du taux de phéromone :

$$\forall i,d \quad \tau_{i,d}^{+} = \tau_{i,d} * (1-\alpha') + \alpha' * \Delta\, taux$$

$\Delta\, Taux = |E_s| * 0.01$ (l'optimum n'est pas connu)

- Le résultat obtenue avec recherche locale pour la 1ere séquence benchmarks avec (nombre de fourmis =40, et les autre paramètres sont fixer par défaut) est égale a «-9 » ,avec l'itération de la 1éreaparition « 85 »et le temps d'exécution « 5.217 » .

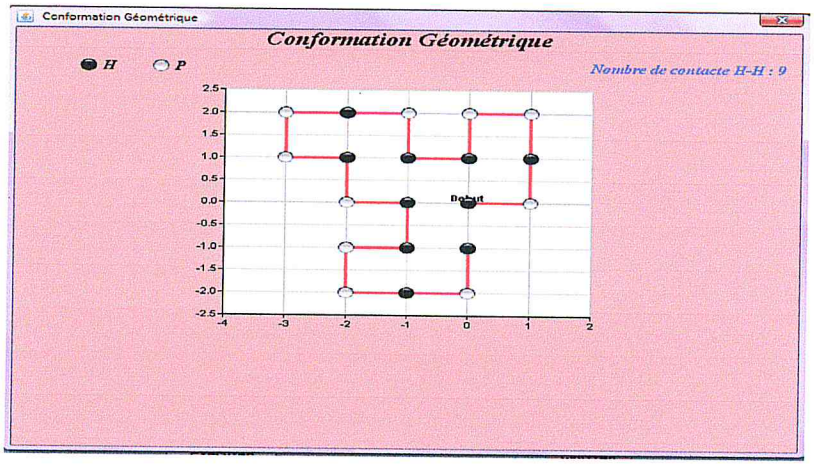


Fig4.10 Conformation trouvée pour la 1ère séquence de benchmarks « aléatoire »

- Le résultat obtenue pour la 2ème séquence benchmarks avec (nombre de fourmis = 40, et les autre paramètres sont fixer par défaut) est égale a «-9 », avec l'itération de la 1ére aparition « 141 »et le temps d'exécution « 1.624».

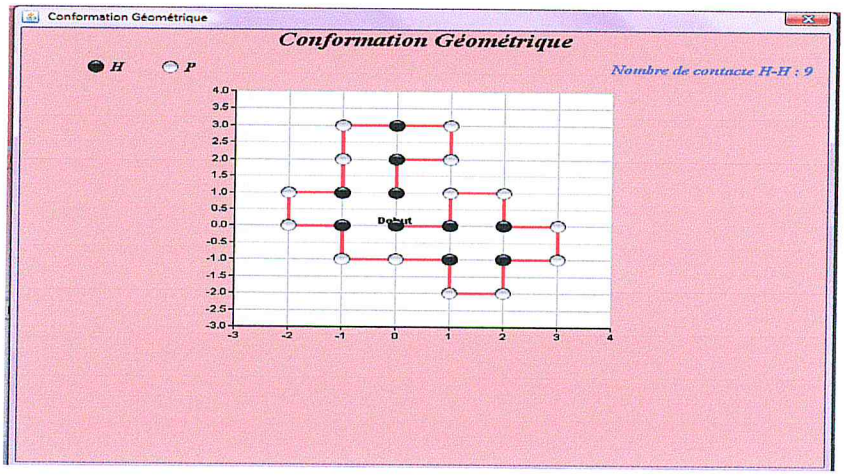


Fig4.11 Conformation trouvée pour la 2ème séquence de benchmarks « aléatoire »

4. Conclusion

On a présenté dans ce chapitre le logiciel dédié a la résolution du problème du repliement de protéine par les colonies de fourmis hybridé avec la recherche locale et on à donné quelques tests les plus significatives.

Notre algorithme s'est montré très efficace pour la résolution de problème de repliement de protéine.

CONCLUSION GENÉRALE

Conclusion générale

La classe des problèmes NP-Complet a toujours suscité un grand intérêt pour les chercheurs vu l'importance dans la pratique des problèmes qu'elle contient. L'utilisation des métaheuristiques, comme les algorithmes de colonies de fourmis (MBO), permet une résolution approchée de ces problèmes, tout en réduisant les délais de calcul exorbitants que coutent les méthodes exactes.

Ces métaheuristiques ne sont vraiment efficaces que si elles sont jumelées avec une des méthodes de recherche locale.

Notre travail consisté à résoudre un problème NP-Difficile à grand importance dans le domaine biomédicale et biotechnologique, étant donnée une protéine, à trouver un repliement de moindre énergie. Notre approche de résolution pour ce problème consisté a proposer des modèles de coopération/hybridation entre deux métaheuristiques une recherche locale et un algorithme de colonies de fourmis.

Notre logiciel est un outil simple à utiliser. Il permet de prédire la structure native d'une protéine à partir de sa séquence d'acides aminés. Il permet à l'utilisateur d'introduire ses données, de choisir ses paramètres et de visualiser les différents résultats obtenus avec et sans recherche locale.

L'obstacle que nous avons rencontré est bien celui du paramétrage de la méthode, et lors de corporation de la recherche locale .En effet, malgré ses nombreux avantages, l'algorithme nécessite un temps très important de tests afin d'en déterminer la configuration paramétrique. Nous avons donc effectué quelques choix afin de procéder à des tests qui nous semblaient intéressants.

Les résultats des tests sont très satisfaisantes, on à pu avoir les optima de plusieurs séquences et hybridation avec la recherche locale à perfectionner les résultats.

Enfin, nous espérons que ce modeste travail, servira comme base et sera bénéfique pour d'autres étudiants, qui voudraient continuer dans ce domaine, et soit complété par d'autres études, qui tiennent compte la résolution des problèmes NP-Difficiles, d'une part, et la résolution du problème du repliement de protéines d'autre part.

BIBLIOGRAPHIE

ACO08 Ameur Khadîdja, Chentir Afef « les colonies de fourmis appliquées au problème du repliement de protéine », proposé par Mme Bensettiti, 2007/2008, Mémoire de Licence informatique.

AG05 Jean-Marc Alliot, Nicolas Durand « Algorithmes génétiques », March 14, 2005.

ANG05 Eric Ange, and Evripidis Bampis, and Malek Rahouelan Hakim Bouchema: «Ants Metaheuristic and Constraint Programming Coorperation for the Protein Folding Problem»,2005.

BA09 Bellag Aicha, Orabi-Salem Fatima-Zohra « les algorithmes génétiques appliquées au problème du repliement de protéine », proposé par Mme Bensettiti, 2008/2009, Mémoire de Licence informatique.

CEDRIC Le Site http://cedric.cnam.fr/AfficheEquipe.

DAP08 ABDOU BADREDINE « deux approches parallèles basées sur les colonies de Fourmies pour la résolution des problèmes de la T-coloration des graphes » 2008/2009.

DIL95 K. A. DILL, S. BROMBERG, K. YUE, K. M. FIEBIG, D. P. YEE, P. D. THOMAS and H. S. CHAN, «Principles of protein folding :A perspective from simple exact models», Protein Science ;4:561-602,1995.

DOC07 HERNANE Soumeya, BELKADI Khaled, « Deux stratégies parallèles de l'optimisation par colonie de fourmis », SETIT 2007, International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, Tunisia.

Dor97 Dorigo M. et L.M. Gambardella. 1997« Ant Colony System: a cooperative leaning approach to the Travelling Salesman Problem», IEEE Transactions on Evolutionary Computation, vole1, pp. 53-66.

DOR99 Marco Dorigo et al. « Ant Colony Optimization: A New Meta-Heuristic», McGraw-Hall, 1999.

FAB06 Fabian TEHEUX, mémoire de licence informatique, Marianne ROOMAN, Esteban ZIMANYI « Algorithme d'optimisation par Colonie de fourmis : développement et application à la Prédiction ab initio de la structure native des protéines », 2006.

JKH99 Jin-Kao Hao, Philippe Galinier, Michel Habib. « Méthaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes », Revue d'Intelligence Artificielle, Vol : No. 1999.

JR08 Jamal Rebaine « NP complétude », cours, 2007.

HAM03 N.HAMMANI, S.GHAROUT, « Résolution des problèmes Max-Sat et partitionnement en utilisant l'optimisation par Colonie de Fourmis », mémoire d'ingéniorat à l'INI, 2002-2003.

HEC03 Jean-François Heche, LIEBLING M. THOMAS, DE WERRA DOMINIQUE, « Recherche opérationnelle pour ingénieurs », Presses Polytechniques et Universitaires Romandes (PPUR), 2003.

HS-97 HS. Chan and KA. Dill. From levinthal to pathways to tunnels. *Nat. Stru.Biol.*, 4:1O.19, 1997.

KIC Le site www.kimonte.com , Algorithme de colonies de fourmis

LIN07 Markus Lindstrome, « protéines : modélisation, prédiction de Structure et conception », département d'informatique, université Libre de Bruxelles, printemps des sciences, 2007.

MBO08 Ramdani toufik,Bendjilali zine el abidine « résolution de problème de repliement de protéine par une approche métaheuristique », Université Houari Boumediene,proposé par M^{r}. A.Boukra,2008 /2009.

Om LITIS Omar gaci « Etude de la dynamique du repliement des protéines » 3juillet 2008, laboratoire LITIS université de havre.

OP08 « optimisation combinatoire » Mathématique, 2008.

OPF06 COSTANZO Andrea, LUONG Thé Van, MARILL Guillaume, « Optimisation par colonies de fourmis »,2006 .

RCS Christine Solnon « Résolution de problèmes combinatoires et optimisation par colonies de fourmis »,

d'implémentation des métaheuristiques.

ZAN89 S. H. Zanakis, J. R. Evans et A. A. Vazacopoulos, « *Heuristic methods and applications: A categorized survey*». European Journal of Operational Research, Vol. 43, p. 88-110, 1989.