Habilitation à Diriger les Recherches

Spécialité : informatique

Présentée devant l'Université de Rennes I

par

Robin Gras

Structure des espaces de recherche, complexité des algorithmes d'optimisation combinatoire stochastique et applications à la bioinformatique

Soutenue le 3 décembre 2004
devant un jury composé de :

Pr. Rumen Andonov
Pr. Antoine Danchin
Pr. Boi Faltings
Pr. Jin-Kao Hao
Dr. Jacques Nicolas
Pr. El-Ghazali Talbi
Pr. Marco Tomassini
Pr. Alain Viari

A Sophie, à mes parents.

Un grand merci à

Antoine Danchin, Boi Faltings et El-Ghazali Talbi pour avoir accepté de rapporter ce travail.

Jacques Nicolas pour m'avoir soutenu depuis le début et porté un intérêt constant pour mon travail.

Rumen Andonov pour sa grande disponibilité et les nombreuses discussions enrichissantes que nous avons pu avoir.

Frédérique Lisacek pour tous ses précieux conseils et son enthousiasme permanent pour de nouveaux projets.

Jin-Kao Hao, Marco Tomassini et Alain Viari pour avoir accepté de participer à mon jury.

Toutes celles et ceux avec qui j'ai travaillé ces dernières années et qui ont été par les innombrables discussions et échanges à la source d'une grande part de ce travail, parmi lesquels je remercie particulièrement David Hernandez, Patricia Hernandez, Yoann Mescam et Markus Müller de m'avoir supporté (dans tous les sens du terme) pour encadrer leurs travaux ainsi que Pavel Dobrokhotov, Julien Frey, Olivier Martin, Gregory Theiller, Nadine Zangger, Amos Bairoch, Christian Pellegrini, Bastien Chopard, Elisabeth Gasteiger, Christine Hoogland, Dominique Lavenier…

Table des matières

CHAPITRE 1

Introduction

1.1 Mon parcours

Mon activité de recherche s'inscrit dans deux grandes thématiques : l'optimisation combinatoire et la bioinformatique. Ces thématiques sont en large mesure complémentaires par le fait qu'une importante partie des problèmes complexes en bioinformatique peut être modélisée sous la forme d'un problème d'optimisation combinatoire. Ce document présente une synthèse des travaux que j'ai réalisés et dirigés ou co-dirigés dans ces deux domaines depuis mon engagement en tant que post-doctorant en 1998 à l'Institut Suisse de Bioinformatique (ISB) puis ma nomination comme responsable de recherche dans ce même institut.

Mon stage de DEA d'informatique (Gras R., 1994), ma thèse (Gras R., 1997) et mon année d'ATER ont été l'occasion pour moi de m'initier à la bioinformatique et à la combinatoire. J'ai ainsi soutenu la première thèse sur un thème bioinformatique de l'université de Rennes et appris bon nombre de concepts en biologie qui m'étaient jusqu'alors inconnus. Le sujet portait sur la conception d'un outil de visualisation et d'analyse des grandes séquences biologiques par l'intermédiaire de leur contenu en mots. Le principe était de construire un index exhaustif de toutes les répétitions exactes contenues dans une ou plusieurs séquences sous la forme d'un arbre des suffixes généralisé puis d'utiliser cette structure à la fois comme un outil de visualisation du contenu en mots et comme un outil de calcul de statistiques sur ces mots. Conjointement, j'ai proposé un algorithme combinatoire de recherche de mots avec erreurs et gaps utilisant cette structure et qui est linéaire sur la taille de la requête. J'ai par la suite utilisé cet outil pour réaliser des études sur les mots particulièrement rares ou fréquents et les sites de régulation de la transcription dans les premiers génomes complets disponibles : *Escherichia coli* et *Bascilus subtilis*.

J'ai été engagé à l'ISB pour un projet de bioinformatique très différent de ce sur quoi j'avais travaillé jusqu'alors puisqu'il s'agissait d'un projet de protéomique. Mon travail a consisté à développer un outil d'identification automatique de protéines à partir de données de spectrométrie de masse. Parallèlement, je me suis intéressé aux méthodes d'optimisation combinatoire stochastiques et plus particulièrement aux algorithmes évolutifs. J'ai réalisé un outil d'identification automatique, SmartIdent, basé sur une approche d'apprentissage par un algorithme génétique dédié. Cet outil, reconnu comme l'un des plus performants parmi ceux disponibles à l'époque, a été intensivement utilisé au Geneva Proteome Center de l'Hôpital Cantonal Universitaire de l'université de Genève (HCUGE).

En tant que responsable de recherche, j'ai dirigé et codirigé de nombreux stages de DEA et plusieurs thèses dont l'une, "Molecular Scanner Data Analysis", a été soutenue par Markus Muëller en février 2003. Ces travaux portent sur l'étude de la complexité dans les problèmes d'optimisation combinatoire, la conception de méthodes efficaces d'exploration d'espace de recherche par des méthodes évolutives. Ils portent aussi sur le développement d'algorithmes métaheuristiques dédiés appliqués à des problèmes bioinformatiques et d'outils de découverte de motifs dans les séquences biologiques, d'alignements locaux et d'identification automatique des protéines. J'ai également été consultant pour la société GeneProt Inc., en tant que coordinateur du groupe chargé de définir le protocole et les outils d'analyse des données de spectrométrie de masse (ces outils visant à identifier et caractériser les protéines de protéomes complets) et en tant que membre du groupe chargé de concevoir le super-calculateur (cluster de machines Alpha) destiné à l'analyse de ces données.

1.2 L'optimisation combinatoire

L'optimisation combinatoire est une problématique consistant, pour un ensemble de variables prenant des valeurs entières, à déterminer leurs instanciations, éventuellement sous contrainte, correspondant au maximum (ou au minimum) d'une fonction de ces variables. La majeure partie des problèmes appartenant à cette classe a été démontrée NP-complet et donc particulièrement difficile à résoudre. Des méthodes exactes (branch and bound, programmation dynamique…) peuvent s'avérer efficaces dans certaines instances de ces problèmes mais le plus souvent aucune solution exacte ne peut être découverte en un temps raisonnable et des approches heuristiques non exactes doivent être utilisées pour proposer des solutions approchées de bonne qualité. Le grand nombre de problèmes pouvant être représentés sous cette forme ou sous une forme similaire (problème d'ordonnancement ou de partitionnement optimal) donne une importance prépondérante à ce domaine de recherche. Il est donc essentiel de disposer de méthodes heuristiques efficaces pour découvrir des solutions approchées et de déterminer les classes de problèmes solubles par chacune de ces méthodes.

Mon travail dans le domaine de l'optimisation combinatoire porte sur différents aspects. Je m'intéresse tout d'abord à la notion de complexité d'un problème donné sachant le paysage (espace de recherche munie d'une fonction de voisinage et auquel est associée, en chaque point, la valeur de la fonction à optimiser) qui lui est associé. Quels sont les paramètres et les conditions qui influencent la complexité d'un problème? Quelles conséquences cela a-t-il sur l'efficacité des différentes méthodes, exactes ou non, utilisées pour le résoudre? Quelle stratégie est-elle plus pertinente pour résoudre un tel problème?

Ma deuxième contribution dans ce domaine se situe dans le cadre des métaheuristiques. Les métaheuristiques sont des algorithmes d'exploration d'espace de recherche heuristiques dont le but est d'être aussi générique que possible. J'étudie plus précisément l'une des classes de ces approches : les algorithmes évolutifs. Leur principe est de réaliser un échantillonnage de l'espace de recherche et d'utiliser l'information extraite de cet échantillon pour explorer l'espace de recherche le plus efficacement possible. Du point de vue des métaheuristiques, on peut voir cela comme une stratégie

d'exploration basée sur la mise en compétition d'un ensemble de solutions potentielles et sur la "coopération" entre ses solutions pour partager de l'information (sous-solutions pertinentes). Ce partage d'information vise à produire de nouvelles solutions de meilleure qualité, c'est-à-dire correspondant à une valeur plus élevée (ou moins élevée si c'est un problème de minimisation) de la fonction à optimiser. Quelle est la meilleure manière de réaliser cette coopération? Comment utiliser l'information extraite de l'échantillon pour déduire la structure du paysage du problème et en tirer parti? Quelles sont les limitations de telles approches? Comment pourraient-elles être couplées à d'autres stratégies pour pallier ces limitations? Quelle est la pertinence d'une parallélisation de l'échantillonnage? Ce sont ces différentes questions auxquelles j'essaierai d'apporter quelques réponses dans le deuxième chapitre de ce document en m'appuyant sur deux travaux de stage (Julien Frey et Olivier Martin) menés sous ma direction, des collaborations avec l'équipe ROI de l'université de Valenciennes et des résultats personnels obtenus récemment.

Parallèlement, je cherche à appliquer ces connaissances pour résoudre des problèmes bioinformatiques concrets. Cette deuxième thématique de ma recherche repose aussi bien sur l'adaptation de techniques d'exploration à ces problèmes bioinformatiques que sur la création de nouvelles approches bioinformatiques pour résoudre des problèmes biologiques réels.

1.3 La bioinformatique

Grâce à de nouvelles technologies performantes (séquençage à grande échelle, spectrométrie de masse, puce à ADN…) la bioinformatique s'est considérablement diversifiée ces dernières années. La quantité de données générées continue, voire accélère, son augmentation exponentielle et rend plus cruciale encore la mise au point d'algorithmes d'analyse performants. La capacité à explorer et à comparer un grand nombre de séquences biologiques est plus que jamais d'actualité. Je poursuis donc cet axe de recherche initié pendant ma thèse en l'orientant plus spécifiquement vers l'analyse des séquences protéiques. Je me suis également orienté vers l'étude d'un nouveau type de données produites dans un nombre croissant de laboratoires : les données de protéomique.

1.3.1 La protéomique

Une part importante de mon activité de recherche à l'ISB est liée à la protéomique. Cette science récente a pour objectif d'étudier les protéomes d'organismes vivants. Le protéome est l'ensemble de protéines exprimées à un moment donné par un organisme ou un tissu donné. Contrairement au génome, l'aspect extrêmement dynamique de l'expression des protéines fait qu'il existe un nombre très important de protéomes différents pour un même organisme. Chaque organisme possède un nombre important de différentes protéines (de quelques milliers pour les prokaryotes jusqu'à quelques dizaines de milliers pour les eukaryotes) auxquelles se rajoutent un grand nombre de variants (modifications post-traductionnelles, mutations, fragments…) en fonction des tissus où ces protéines sont exprimées et des conditions environnementales.

Cette diversité complique fortement la tâche de l'analyse des données protéomiques et rend indispensable la conception d'outils informatiques à la fois robustes et rapides.

Le processus classique d'analyse protéomique comprend généralement cinq étapes : la séparation des protéines d'un échantillon à analyser (par électrophorèse bidimensionnelle par exemple), la digestion des protéines par un ou plusieurs enzymes produisant un ensemble de peptides, la mesure de la masse de ces peptides ou de leurs fragments par spectrométrie de masse et la comparaison de ces masses avec des bases de données de séquences protéiques ou génomiques traduites. Ces masses appariées sont utilisées pour identifier les protéines de l'échantillon ou leurs éventuels variants. La découverte de la fonction biologique de nouvelles protéines, variants de protéines ou complexes protéiques repose sur plusieurs sources d'information (données expérimentales d'expression, séquences, annotations dans les bases de données, littérature…). Cette approche doit être automatisée au maximum pour permettre d'analyser en temps réel l'énorme quantité de données produites dans les environnements expérimentaux produisant des données à haut débit comme on en rencontre couramment maintenant.

Notre travail consiste ici à développer de nouveaux algorithmes permettant d'identifier automatiquement des protéines à partir de données de spectrométrie de masse en ayant la possibilité de découvrir de nouveaux variants jusqu'alors non annotés. Nous avons suivi une approche basée sur un apprentissage de fonction d'identification par algorithmes évolutifs et sur des techniques de clustering. Ceci a été réalisé au cours de mon post-doctorat puis de deux thèses (Markus Muëller et Patricia Hernandez) sous ma direction. Une première approche de prédiction de fonction de protéines par programmation génétique a également été réalisée pendant le stage de Julien Frey. Je présente ces travaux dans le troisième chapitre de ce document.

1.3.2 L'analyse de séquences biologiques

L'analyse de séquences biologiques a pour but de déterminer *in silico* et le plus automatiquement possible des propriétés fonctionnelles d'une macromolécule biologique directement à partir de sa séquence d'acides nucléiques (dans le cas de séquences génomiques) ou d'acides aminés (dans le cas des protéines).

Au cours de l'évolution, les séquences d'ADN sont sujettes à des mutations. Ces mutations peuvent avoir des conséquences très différentes suivant l'endroit où elles se produisent. Une mutation qui n'affecte pas la survie d'un organisme est susceptible d'être conservée et transmise aux générations suivantes alors qu'une mutation sur un site de régulation par exemple risque d'avoir des conséquences dramatiques pour la survie de l'organisme et donc de disparaître avant d'avoir été transmise. On considère ainsi que les sites intervenant dans des fonctions biologiques doivent se conserver à travers les générations. En recherchant des similarités entre séquences on espère découvrir de tels sites. Les principales techniques utilisées consistent en la comparaison de deux ou plus séquences par des algorithmes d'alignements, la découverte de zones de forte similarité (sites biologiques) dans un ensemble de séquences ou la recherche d'un motif biologique déjà connu dans de nouvelles séquences.

La structure tridimensionnelle des macro-molécules biologiques et plus particulièrement celle des protéines détermine en grande partie leur fonction. Cette

structure est contrainte par des interactions physico-chimiques entre molécules élémentaires (les acides aminés pour les protéines). Ces interactions sont connues mais il est très difficile de prédire leur implication dans le repliement tridimensionnel de la molécule directement à partir de la séquence linéaire. Les motifs utilisés pour caractériser une famille de protéines (ensemble de protéines participant à une même fonction) sont le plus souvent dérivés d'analyses linéaires des séquences et par conséquent ne représentent que des conservations locales au sein de ces séquences. Ce type de motif n'est alors pas à même de représenter des similarités basées sur la conservation d'interactions physico-chimiques entre régions distantes d'une même séquence. Il est ainsi nécessaire de concevoir une représentation de cette conservation distante de façon à pouvoir découvrir automatiquement des régions fonctionnelles de par leur structure.

Nous menons deux études sur les approches de découverte de motifs. Dans une première étude, travail de thèse de David Hernandez sous ma direction, nous avons développé un algorithme de découverte de motifs par alignement multiple local et une approche hybride évolutionniste et de recherche locale. Dans la deuxième, travail de thèse de Yoann Mescam co-dirigé par Jacques Nicolas de l'INRIA de Rennes et par moi-même, nous nous intéressons à la découverte de zones de conservation d'interactions distantes en définissant une nouvelle mesure de covariation et en étendant l'approche précédente à la découverte de multiples zones de conservations locales. Nous utilisons ensuite ces zones comme des générateurs de zones potentiellement covariées. Ces deux travaux seront présentés plus en détail dans le troisième chapitre de ce document.

CHAPITRE 2

Complexité des paysages et exploration par échantillonnage

La notion d'optimisation est une notion très répandue du fait que la résolution dans grand nombre de problèmes concrets nécessite, à un moment ou un autre du processus de résolution, d'optimiser un ensemble de paramètres (ou variables) vis-à-vis d'une *fonction objectif* donnée (fonction de *fitness*). Ces variables peuvent prendre des valeurs réelles ou entières. Les problèmes de ce type dans lesquels les variables ne peuvent prendre que des valeurs entières sont appelés problèmes d'*optimisation combinatoire* et ce sont ceux qui nous concerneront dans ce mémoire. L'ensemble des combinaisons pouvant être prises par ces variables constitue l'*espace de recherche* du problème. *L'optimisation combinatoire sous contrainte* définit un ensemble de contraintes sur les variables rendant non valide un sous-ensemble de l'espace de recherche. Nous ne considérerons pas dans ce mémoire les problèmes d'optimisation combinatoire sous contrainte mais uniquement les problèmes d'optimisation combinatoire globale. Bien que les méthodes que nous présenterons peuvent s'appliquer aux problèmes avec contraintes moyennant un certain nombre d'adaptations, elles ne sont pas spécifiquement adaptées pour eux. Bon nombre des concepts sur la complexité et la structure des espaces de cherche présentés dans ce mémoire sont toutefois directement applicable à l'optimisation combinatoire sous contrainte. Cet espace de recherche ordonné suivant un *voisinage* et auquel on associe en chaque point la valeur de la fonction de fitness pour l'instanciation correspondante des variables est appelé *paysage* du problème. Nous nous intéresserons dans ce chapitre à deux problématiques de l'optimisation combinatoire : d'une part le lien entre la structure du paysage et la complexité du problème à résoudre et d'autre part l'efficacité des *métaheuristiques* dites *évolutives* utilisant un échantillonnage statistique biaisé de l'espace de recherche pour son exploration. Par souci de simplification des notations nous n'évoquerons pas ici les problèmes d'optimisation combinatoire consistant en un ordonnancement (par exemple le problème du voyageur de commerce) ou un partitionnement (par exemple un problème de clustering) optimal des variables. Nous nous limiterons également aux problèmes de maximisation constitués uniquement de variables binaires. Tout ce qui sera présenté dans ce chapitre peut toutefois être étendu facilement aux autres problèmes d'optimisation combinatoire.

2.1 Définitions générales

Soit $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ un ensemble de n variables binaires et $C = 2^{\mathcal{X}}$ l'ensemble des parties de $\mathcal{X}$. On note X un élément de C. Soit F une fonction de C dans R. Un problème d'optimisation combinatoire O est un problème qui cherche le ou les maximums de la fonction objectif $F(X)$ (aussi couramment appelée fonction de fitness) pour tous les points X de l'espace de recherche C. On dit que O est un problème de taille n. Sa complexité est le nombre minimum de points X de C pour laquelle $F(X)$ doit être calculée pour garantir de découvrir le ou les maximums. Ce nombre est dépendant de n mais également de bien d'autres paramètres dont l'influence peut être très supérieur à celle de n.

Soit $M = \{X_m \in C \mid \forall X_i \in C,\ F(X_m) \geq F(X_i)\}$ l'ensemble des points solutions de O (appelés également maximums globaux), c'est-à-dire l'ensemble des points de C pour lesquels F est maximale.

On appelle métaheuristique un algorithme générique qui "explore" un sous-ensemble Ex de C et produit en résultat l'ensemble des points $S = \{X_s \in Ex \mid \forall X_i \in Ex\ F(X_s) \geq F(X_i)\}$. Les problèmes considérés étant le plus souvent NP-complets, on ne peut garantir de trouver la solution en un temps polynomial sur n. Les métaheuristiques ne sont en général pas des algorithmes exacts et n'offrent pas de garantie de trouver une solution en un temps "raisonnable". Ils cherchent plutôt à trouver une solution sous-optimale de "bonne qualité" en un temps raisonnable. L'efficacité d'un tel algorithme dépend donc en grande partie des rapports entre $|Ex|$ et $|C|$, ce rapport devant être le plus petit possible, et entre $F(X_s)$ et $F(X_m)$, ce rapport devant être le plus grand possible.

Les métaheuristiques explorent C en utilisant un ou plusieurs opérateurs qui, à partir d'un ou plusieurs points de C, vont produire un ou plusieurs points de C. Soit Ec un sous-ensemble (ou échantillon) de C. On définit un opérateur o comme étant une fonction de Ec_1 dans Ec_2. On dit que Ec_2 est l'ensemble des points de C voisins des points de Ec_1 pour o. On définit une fonction $\mathcal{V}o$, dite de voisinage de l'opérateur o, tel que $\mathcal{V}o(Ec_1) = Ec_2$. On appelle $P = \{C, F, \mathcal{V}o\}$ un paysage de O pour l'opérateur o. Un paysage est donc dépendant du voisinage associé à un opérateur.

On peut dès lors définir la notion de maximum local qui est dépendante du voisinage et donc d'un opérateur. L'ensemble des maximums locaux pour l'opérateur o est l'ensemble $Mo = \{X_m \in C \mid \forall X_i \in \mathcal{V}o(X_m),\ F(X_m) \geq F(X_i)\}$.

On définit le *bassin d'attraction* d'un maximum suivant un opérateur comme l'ensemble $Bo(X_m) = \{X_0 \in C \mid \exists X_1, \ldots, X_n, X_n = X_m \in Mo$ et $X_{i+1} \in \mathcal{V}o(X_i)\ \forall 0 \leq i \leq n\}$, c'est-à-dire l'ensemble des points à partir desquels on peut atteindre le maximum en appliquant l'opérateur o.

On appelle *grimpeur strict* un type de métaheuristiques qui n'utilise que des opérateurs de la forme $o : Ec_1 \rightarrow Ec_2$ avec $\forall X_i \in Ec_1$ et $X_j = o(X_i)$, $F(X_j) \geq F(X_i)$ et $X_j \in \mathcal{V}o(X_i)$. On appelle *grimpeur stochastique* un type de métaheuristiques qui n'utilise que des opérateurs de la forme $o : Ec_1 \rightarrow Ec_2$ avec $\forall X_i \in Ec_1$, $X_j \in \mathcal{V}o(X_i)$ et $X_j = o(X_i)$, $P(F(X_j) \geq F(X_i)) = g(\mathcal{V}o(X_i))$ où g est une fonction non uniformément distribuée sur [0..1]

avec un biais vers les valeurs élevées. En général et sauf indication contraire, on a aussi $|Ec_1| = |o(Ec_1)| = 1$. Le voisinage le plus couramment utilisé pour les métaheuristiques de type grimpeur est le voisinage de Hamming. Dans ce cas, le voisinage d'un point $X_i \in C$ est $\mathcal{V}o(X_i) = \{X_j \in C \mid H(X_i,X_j) = 1\}$ avec $H(X_i,X_j)$ la distance de Hamming entre X_i et X_j.

On appelle *métaheuristique évolutive*, une métaheuristique qui maintient un échantillon (appelé couramment *population*) *Ec* de points de *C* et utilise un opérateur *os*, dit de *sélection*, qui à partir de *Ec* produit un autre échantillon biaisé *Ec'* tel que $\mathcal{V}os(Ec) = Ec$ et $\forall X_j \in Ec' = os(Ec)$ et $\forall X_i \in Ec$, $X_j \in \mathcal{V}os(Ec)$ et $P(F(X_j) \geq F(X_i)) = g(\mathcal{V}os(Ec))$ où g est une fonction non uniformément distribuée sur [0..1] avec un biais vers les valeurs élevées. A chaque point de *Ec'* sont ensuite appliqués probabilistiquement 0, 1 ou plusieurs autres opérateurs. On ré-applique récursivement ce même processus à la nouvelle population ainsi formée jusqu'à arrêt de l'algorithme.

2.2 Etat de l'art

2.2.1 Complexité

De nombreux travaux ont été menés pour étudier la complexité des problèmes d'optimisation combinatoire. Dans un premier temps, ce sont principalement des preuves de NP-complétude qui ont été réalisées pour certains problèmes classiques (sac à dos, le coloriage de graphe, la recherche d'une clique de taille *k*…) (Garey M.R. and Johnson D.S., 1979; Papadimitriou C.H. and Steiglitz K., 1998). Ce n'est cependant pas un indice de complexité très informatif. On sait que pour une classe de problèmes démontrée NP-complète il existe dans la plupart des cas des instances de ces problèmes solubles en temps polynomial. Malheureusement, les propriétés des problèmes qui les rendent plus ou moins faciles à résoudre sont souvent mal connues. Cette information est d'autant plus importante à découvrir qu'elle est en générale la clé permettant la conception d'un algorithme de résolution efficace. Des notions comme la décomposabilité d'un problème ou le nombre de solutions du problème ont une implication forte sur sa complexité. La complexité d'un problème est aussi bien évidemment directement liée à l'efficacité des algorithmes cherchant à le résoudre et à leur capacité à exploiter ses propriétés (Jones T., 1995).

2.2.1.1 Influence du nombre de maximums

La première propriété influant sur la complexité est le nombre de maximums de *F* (Kallel L. et al., 2001). Par exemple, lorsque $|M|$ devient grand par rapport à $|C|$, un algorithme évaluant au hasard des points de *C* et qui a donc une complexité en moyenne de $|M|/|C|$, devient très efficace. A l'extrême, le problème de "l'aiguille dans une botte de foin" (needle-in-a-haystack) (Forrest S. and Mitchell M., 1993; Goldberg D.E., 1989a), dans lequel $F(X)$ est identique pour 2^n-1 points de *C* et vaut une valeur supérieure pour le dernier point, est un problème de complexité maximale puisque aucune propriété ne peut être exploitée. Dans ce cas, aucun algorithme ne peut faire mieux en moyenne que le

parcours systématique de *C*. Cette notion est indépendante du paysage puisque le nombre de maximums globaux est indépendant de la ou des fonctions de voisinage choisies. Le nombre de maximums locaux est lui totalement lié à la fonction de voisinage (voir définitions). Pour une fonction de voisinage donnée (et donc pour un opérateur d'exploration donné), le paysage présente un nombre fixe de maximums locaux (si ce nombre est supérieur à un, on dit que la fonction est *multimodale*). On estime en général que plus |*Mo*| est grand, mais aussi plus le rapport |*Mo*|/|*M*| est grand, plus le problème est difficile (Palmer R., 1991; Stadler P., 1995). Une propriété des paysages plus précise que le seul nombre de maximums est le nombre et la taille des bassins d'attractions. Plus le bassin d'attraction d'un maximum suivant un opérateur *o* est grand et plus la probabilité de découvrir ce maximum en partant d'un point aléatoire de *C* et en utilisant un algorithme basé sur *o* est élevé. Ainsi la taille respective des bassins d'attraction des maximums locaux et globaux dans un paysage donné a une forte implication sur la complexité de l'exploration de ce paysage et donc sur l'efficacité de la méthode de recherche correspondante.

Des études ont été menées pour analyser cette implication en utilisant la technique de *grimpeur-inverse* (reverse hillclimbing, voir figure 1) (Jones T., 1995). Le principe est de déterminer les bassins d'attraction des maximums d'un paysage en appliquant de façon inverse l'opérateur à partir des maximums dans toutes les directions possibles. L'ensemble des points ainsi atteints constitue le bassin d'attraction de chaque maximum. Ce concept a été étendu aux opérateurs de grimpeurs stochastiques débouchant sur le calcul des probabilités d'atteindre un maximum donné à partir d'un point de *C* donné. Ces méthodes permettent d'apporter une information très riche sur la structure du paysage et donc sur la complexité du problème. Elles sont cependant fortement limitées en ce qui concerne la taille des problèmes auxquels elles sont applicables du fait de l'énorme quantité de calculs nécessaire à la détermination des bassins d'attractions. Il peut toutefois être intéressant de faire ce genre de calcul pour de petites instances d'une classe de problèmes pour essayer de tirer des propriétés intrinsèques à cette classe et donc extrapolables à de plus grandes instances. Une approche basée sur un échantillonnage des bassins d'attractions pour évaluer la stagnation de l'exploration d'un algorithme de recherche par grimpeur a été donnée dans (Cotta C. et al., 1999), conduisant à un bon estimateur de terminaison de l'algorithme.

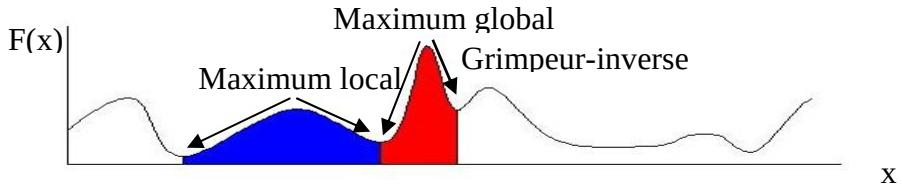


Figure 1. Bassin d'attraction d'un maximum global (rouge) et d'un maximum local (bleu) découvert par grimpeur-inverse.

2.2.1.2 Décomposition et épistasie

On s'intéresse ici au cas où la fonction objectif F peut être décomposée en plusieurs sous-fonctions. Dans le cas où la fonction F de taille n peut être décomposée en n/k sous-fonctions de taille k indépendantes, la complexité n'est plus de 2^n mais de $2^k \cdot n/k$ (si aucune autre propriété du problème ne peut être exploitée réduisant encore sa complexité) (Jansen T., 2001). Dans la situation extrême où les n variables sont indépendantes les unes des autres, la complexité est linéaire en n. Le théorème "no free lunch" (Wolpert D. and Macready W., 1997) ayant montré que la performance de n'importe quel algorithme de recherche moyenné sur tous les paysages possibles est la même que celle d'un algorithme de recherche aléatoire, il est illusoire d'espérer trouver un algorithme générique résolvant efficacement tous les problèmes d'optimisation combinatoire. Il est plus important alors de découvrir quel algorithme choisir ou construire pour un problème donné avec ses propriétés spécifiques. Un algorithme de programmation dynamique, par exemple, exploite parfaitement la propriété de décomposition d'un problème (complexité en $2^k \cdot n/k$), pour peu que l'on soit capable d'ordonner les variables de façon à ce que celles participant à la même sous fonction soient situées les unes à coté des autres dans la représentation du problème (Weinberger E.D., 1996). De la même façon, l'ordre dans lequel les variables vont être instanciées dans un algorithme de *branch and bound* et la capacité de l'heuristique de coupure à exploiter des propriétés du problème vont avoir une influence majeure sur l'efficacité de l'approche. Ainsi la décomposabilité d'un problème joue un rôle considérable sur sa complexité. Cette notion est très similaire à celle de l'épistasie (Davidor Y., 1990; Reeves C. and Wright C., 1995).

Le degré d'épistasie d'un problème correspond au nombre maximum de variables dont dépend, dans le calcul de F, chacune des n variables. Par exemple, un problème d'épistasie zéro est un problème où toutes les variables sont indépendantes[1]. L'épistasie est un concept plus général que celui de décomposabilité puisqu'un problème de degré d'épistasie k n'implique pas que le problème soit décomposable en n/k sous-problèmes complètement indépendants. En effet, les sous-problèmes peuvent contenir des variables communes et on peut, par exemple, tout à fait avoir n sous-problèmes de taille k. k variables dépendantes sont appelées un *bloc*. Les blocs peuvent être ou non chevauchants s'ils partagent des variables. Un exemple d'un problème classique de taille n, d'épistasie k et avec chevauchement est le *NK-landscape* (Kauffman S.A., 1989; Kauffman S.A., 1993). L'idée du NK-landscape est de pouvoir construire très facilement des problèmes de difficulté et de niveau d'épistasie variable. Un NK-landscape est un problème d'optimisation combinatoire de taille n et d'épistasie k (chaque variable est dépendante de k autres variables). La fonction F est définie par la somme de n sous-fonctions F_i (une pour chaque variable x_i). La valeur $F_i(x_i)$ est elle-même déterminée par l'état de x_i et des k variables dont elle dépend (on appelle cet ensemble de k variables, c_i le *contexte* de x_i). La fonction F est construite aléatoirement en choisissant au hasard pour chaque x_i les k variables dont elle dépend, c'est-à-dire son contexte c_i, et en construisant également aléatoirement la table qui donne la valeur des $F_i(x_i)$ en fonction de l'état de leur contexte

[1] Le problème onemax, dans lequel on cherche le maximum de $F(X)$ qui est égal à la somme des variables X, est un problème classique d'épistasie zéro.

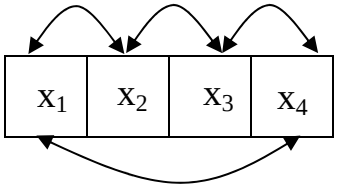
(une table de taille $n \cdot 2^{k+1}$ correspondant à n fonctions F_i). Par exemple (issu de (Jones T., 1995)), dans le cas où $n = 4$ et $k = 2$, on pourrait avoir la situation suivante :

variables

Etat de c_i	x_1	x_2	x_3	x_4
0**0**0	.18	.04	.73	.91
0**0**1	.42	.77	.35	.11
0**1**0	.68	.31	.04	.89
0**1**1	.91	.17	.25	.70
1**0**0	.93	.12	.73	.53
1**0**1	.59	.64	.82	.77
1**1**0	.19	.94	.38	.21
1**1**1	.43	.49	.62	.55

fonction F_4

contexte des variables



Calcul de $F(0110)$

variable	état de c_i	valeur de $F_i(x_i)$
1	0**0**1	.42
2	0**1**1	.17
3	1**1**0	.38
4	1**0**0	.53

$F(X) = (.42 + .17 + .38 + .53) / 4 = .375$

Dès que k est égal à deux et que le problème n'est pas décomposable en problèmes indépendants (une même variable est contenue dans plusieurs contextes différents), il a été démontré que le problème est NP-complet[2] (Thompson R.K. and Wright A.H., 1996). Des extensions au modèle de NK-landscape ont été proposées pour intégrer le concept de *paysage neutre* (neutral landscape). Deux versions existent : le *NKp-landscape* (Barnett L., 1998) et le *NKq-landscape* (Newman M. and Engelhardt R., 1998). Dans la première version, chaque point X_i de C a une probabilité p que $F(X_i) = 0$. Dans la deuxième, la fonction F est discrétisée en q valeurs distinctes, projetées dans l'intervalle [0, 1]. Dans les deux cas, l'idée est de construire des plateaux dans le paysage, c'est-à-dire des zones dans lesquelles la valeur de F est identique. Des travaux

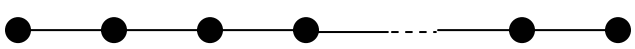
[2] Le problème peut en effet être transformé en MAX-2-SAT qui est NP-complet.

visant à comparer la structure des paysages produits par les trois modèles ont été menés montrant la formation de plateaux pour les NKq-landscapes et une structure plus irrégulière pour les NKp-landscapes (Geard N. et al., 2002). Il est donc clair que ces deux types de modèles produisent des paysages plus complexes à explorer par des opérateurs de voisinage, du fait que ces zones neutres ne permettent pas de faire des choix pertinents quant à la direction à donner à l'exploration. Quand *p* croît vers 1 ou que *q* décroît vers 1 les paysages correspondants ont une structure qui se rapproche de celle des problèmes de type "aiguille dans une botte de foin". Dans ce cas, comme nous l'avons vu précédemment, aucune stratégie d'exploration ne peut faire mieux en moyenne qu'une exploration aléatoire. On peut alors se demander si ces deux modèles sont réellement pertinents pour mieux comprendre le lien entre la structure d'un paysage et la difficulté de son exploration.

L'épistasie peut également être interprétée comme le degré de non-linéarité d'une fonction (Kallel L. et al., 2001). En effet, le degré d'épistasie correspond à la taille des réseaux de dépendances entre variables et donc au nombre de variables impliquées dans une sous-fonction de la fonction principale *F*. Ce sont ces sous-fonctions qui provoquent la non-linéarité de *F*. Un exemple classique est le problème appelé modèle Sherrington-Kirkpatrick (SK) dans la théorie des glasses de spin (Sherrington D. and Kirkpatrick S., 1975). Ce problème consiste à optimiser la fonction :

$$F(X) = -\sum_{1 \le i < j < n} J_{ij} x_i x_j - \sum_{1 \le i \le n} h x_i$$

Avec les J_{ij} des variables aléatoires indépendantes de moyenne zéro et de variance $1/n$, h une valeur réelle et $x_i \in \{-1, 1\}$. Ce problème est non-linéaire de degré deux et est démontré NP-complet dans son cadre général. Pourtant il peut s'avérer simple pour certaines instances. Par exemple, si $J_{ij} = 0$ sauf pour $j = i + 1$, le problème est soluble linéairement. C'est donc la valeur des coefficients qui va influer sur la complexité du problème en déterminant quelles sont les variables qui dépendent les unes des autres et à quel point certaines interactions (couple produit de variables) vont être prépondérantes et en contradiction avec d'autres interactions (changement de signe du produit). On peut alors parler de graphe d'interaction entre variables, dans lequel les nœuds sont les variables et les arcs joignent les nœuds pour lesquels il existe une interaction entre les variables dans la fonction *F*. Ce graphe peut être non-valué, auquel cas seule la présence ou l'absence de lien est importante (il n'y a pas de pondération des interactions), ou valué et dans ce cas chaque arc porte le poids de l'interaction. Dans l'exemple ci-dessus ($J_{ij} = 0$ sauf pour $j = i + 1$), le graphe d'interaction est particulièrement simple puisse qu'il s'agit d'une chaîne :



On peut faire le rapprochement entre un problème SK présenté sous cette forme et d'autres problèmes classiques d'optimisation. Dans l'exemple bien connu du coloriage de graphe, le problème est directement codé sous la forme de son réseau d'interaction, les variables étant les nœuds auxquels on a instancié une couleur. Il est bien connu pour ce

type de problème que la topologie et la densité du réseau influent fortement sur sa complexité (Hogg T. et al., 1996).

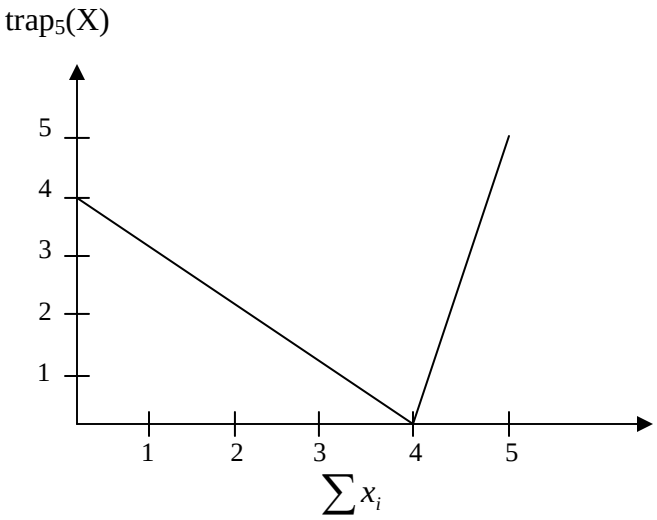
2.2.1.3 Les fonctions trompeuses

Les fonctions *trompeuses* (*deceptive* ou *trap* function) sont des fonctions qui ont été construites dans le but d'être particulièrement difficiles pour les algorithmes grimpeurs. Ackley propose l'une des ces fonctions et montre qu'elle produit un nombre de maximums locaux exponentiel (pour un voisinage de Hamming) (Ackley D.H., 1987). Elles ont depuis été et sont toujours intensivement étudiées et servent fréquemment pour la validation d'algorithmes de recherche (Goldberg D.E., 2002; Martin O. et al., 2003; Pelikan M., 2002; Sastry K. et al., 2004; van Nimwegen E. et al., 1997). On peut les voir en fait comme un cas particulier du problème du nombre de maximums locaux. Or, les maximums locaux étant par définition liés au voisinage choisi, la notion de fonction trompeuse est, elle aussi, liée au voisinage. On peut toutefois donner une autre définition de fonction trompeuse plus informelle. Si on considère l'ensemble des valeurs possibles de $F(X)$ pour tout $X \in C$ et qu'on les trie par valeurs croissantes on obtient un paysage entièrement ordonné par les valeurs de la fonction. Si on exclut de cet ensemble de points les points de M, c'est-à-dire les maximums globaux, et que l'on nomme C' ce nouvel ensemble, on peut donner la définition informelle suivante de fonction trompeuse : une fonction trompeuse est une fonction pour laquelle l'instanciation des variables x_i des points appartenant au sous-ensemble des points de C' de valeurs les plus élevées[3] est la plus différente de l'instanciation des variables x_i des points de M. Dans le cas de problème binaire, la notion de "différente" correspond en fait à la distance de Hamming. Cette définition, bien que liée au voisinage de Hamming, ne fait pas intervenir la notion de maximum local et est donc presque indépendante du paysage choisi. On peut ainsi construire des fonctions intrinsèquement complexes puisque cette définition de complexité ne fait pas directement référence à une fonction de voisinage. Parmi les fonctions trompeuses classiques on peut citer la fonction $trap_5$ (Ackley D.H., 1987) de Ackley ou la fonction *royal road* de Mitchell, Forrest et Holland (Mitchell M. et al., 1992). La première est une fonction de taille 5 définie comme suit :

$$trap_5(X) = \begin{cases} 5 & si \ \sum x_i = 5 \\ 4 - \sum x_i & si \ \sum x_i < 5 \end{cases}$$

La taille de cette fonction étant petite, elle est souvent utilisée comme bloc constitutif d'une fonction de plus grande taille. En général on construit une fonction F décomposable en une somme de plusieurs fonctions $trap_5$. On peut également bien sûr faire varier la taille de la fonction trap. On peut donner une représentation graphique de cette fonction qui illustre clairement sa difficulté pour les algorithmes grimpeurs utilisant le voisinage de Hamming.

[3] On ne spécifie pas la taille de ce sous-ensemble mais plus elle est grande et plus la fonction est trompeuse.



La fonction royal road est une fonction de taille $8 * l$ définie comme suit :

$$R(X) = \sum_{i=0}^{l-1} \prod_{j=1}^{8} x_{8i+j}$$

On peut donc directement relier la complexité de ces fonctions au concept d'épistasie puisque les sous-fonctions représentent une interaction entre les variables qui les composent et qu'elles sont indépendantes entre elles, leur taille correspond au degré d'epistasie du problème. Pour ces fonctions le degré d'épistasie est souvent appelé l'*ordre* de la fonction. On peut introduire une nouvelle propriété des fonctions décomposables qui s'applique bien à ce type de fonctions. La numérotation des variables impliquées dans une même sous-fonction est importante pour certains algorithmes d'exploration comme nous le verrons plus tard. Nous appellerons fonctions *à blocs contigus* les fonctions composées de sous-fonctions dans lesquelles les variables impliquées dans une même sous-fonction auront des numérotations contiguës[4] et fonctions *à blocs non contigus* celles pour lesquelles cette numérotation ne sera pas contiguë[5]. On utilisera le terme *topologie du réseau d'interaction* pour désigner la manière dont sont disposées, les unes par rapport aux autres, les variables dépendantes.

2.2.1.4 Mesures de complexité

Plusieurs mesures ont été crées pour tenter de quantifier la complexité d'un problème. Du fait de la forte corrélation entre le degré d'épistasie et la complexité, certaines d'entre elles sont en fait un calcul de ce degré. La mesure de la *variance d'épistasie* (Davidor Y., 1991; Reeves C. and Wright C., 1995) est définie comme suit :

[4] Les variables x_i, x_{i+1}, x_{i+2},, x_{i+3} et x_{i+5} participent à une même sous-fonction $trap_5$ par exemple.
[5] Les variables x_{2i}, $x_{2(i+1)}$, $x_{2(i+2)}$, $x_{2(i+3)}$ et $x_{2(i+4)}$ participent à une même sous-fonction $trap_5$ par exemple.

$$\frac{\sum\left(F(X)-\sum effets\_lineaires\right)^2}{\sum\left(F(X)-\overline{F}\right)}$$

Avec les sommes extérieures faites pour tous les points de l'espace de recherche, $\overline{F}$ la moyenne des valeurs de *F* pour tous les points de l'espace de recherche et les effets linéaires calculés comme la somme normalisée de la participation de chaque variable indépendamment des autres à la valeur de *F*. Cette mesure utilise le lien entre la notion d' épistasie et la non-linéarité. Cette mesure nécessite toutefois de calculer la valeur de *F* en tout point de l'espace et n'est donc applicable que lorsque la taille des problèmes est particulièrement petite. Une variante de cette mesure est la *bit-wise epistasis*, qui est calculée pour chaque variable et représente un degré de non-linéarité associé à cette variable. Elle nécessite également pour chaque variable un calcul de complexité exponentielle avec le nombre d'autres variables.

La mesure de *corrélation de distance de fitness* (fitness distance correlation) définie par Jones et Forrest (Jones T. and Forrest S., 1995) ne repose pas sur le degré d'épistasie mais sur le lien entre la distance entre les points du paysage et le maximum global le plus proche d'eux et la valeur de la fonction en ces points. Le calcul de cette distance nécessite de spécifier un voisinage, le plus souvent un voisinage de Hamming, et est donc fortement dépendant de ce choix. Si on appelle D_i la distance du point X_i à son plus proche maximum global pour le voisinage choisi, $\overline{D}$ la moyenne de cette distance pour tous les points de l'espace de recherche et $\overline{F}$ la moyenne des *F(X)* pour tous les points de l'espace de recherche, la corrélation de distance de fitness *r* est définie par :

$$C_{FD} = \frac{\sum_{i=1}^{2^n}\left(F(X_i)-\overline{F}\right)\left(D_i-\overline{D}\right)}{2^n}$$

Et par :

$$r = \frac{C_{FD}}{S_F \times S_D}$$

avec S_F et S_D les déviations standards de *F* et *D*. Plus la valeur absolue de *r* est élevée et plus il existe une forte corrélation entre la position par rapport à un maximum global et la valeur de la fonction objectif, donc plus la structure de l'espace est régulière et exploitable par un algorithme grimpeur et donc plus la fonction est considérée comme facile. Le calcul de *r* nécessite d'évaluer tous les points de l'espace de recherche et est donc de complexité exponentielle en fonction de *n*. On peut cependant en faire une estimation par échantillonnage mais il faut tout de même connaître tous les maximums globaux du problème ce qui est le plus souvent irréaliste. Altenberg a de plus montré que cette mesure pouvait mal classifier des problèmes (Altenberg L., 1997). Il a proposé une fonction pour laquelle une approche grimpeur avec un voisinage de Hamming trouve le maximum global en $O(n^2)$ opérations. Cette fonction devrait donc être considérée comme

facile mais la valeur absolue obtenue pour *r* est faible et elle est donc classée comme difficile. Ceci n'est pas étonnant puisque la mesure est dépendante du voisinage choisi. Sachant cela il est relativement facile de trouver des exemples de problèmes pour lesquels un voisinage donné, et en particulier celui de Hamming, conduit à une classification erronée.

Une nouvelle mesure a été définie récemment par Vérel (Vannechi L. et al., 2004; Verel S. et al., 2003) : les *nuages de fitness* (fitness clouds). Son principe est le suivant : on définit un voisinage $\mathcal{V}o$ dans l'espace de recherche *C*; on trace le graphe ayant pour abscisses les valeurs de *F(X)* pour tout $X \in C$ et pour ordonnées $F(\mathcal{V}o(X))$; on discrétise les deux dimensions en deux ensembles d'intervalles $\{F_1, F_2,\ldots, F_m\}$ et $\{F_{Vo1}, F_{Vo2},\ldots, F_{Vom}\}$ et on calcule deux ensembles de valeurs associées $\left\{\overline{F_1}, \overline{F_2},\ldots, \overline{F_m}\right\}$ et $\left\{\overline{F_{Vo1}}, \overline{F_{Vo2}},\ldots, \overline{F_{Vom}}\right\}$ avec $\overline{F_i}$ la valeur moyenne des *F(X)* pour tout $X \in F_i$ et $\overline{F_{Voi}}$ la moyenne des $F(\mathcal{V}o(X))$ pour tout $X \in F_i$. On calcule la pente P_i des segments reliant deux intervalles consécutifs par :

$$P_i = \frac{\overline{F_{Voi+1}} - \overline{F_{Voi}}}{\overline{F_{i+1}} - \overline{F_i}}$$

Et on définit le *coefficient de pente négative nsc* par :

$$nsc = \sum_{i=1}^{m-1} c_i$$

Avec :

$$\forall i \in [1, m-1] \;\; c_i = \begin{cases} P_i \;\; si \;\; P_i < 0 \\ 0 \;\; si \;\; P_i \geq 0 \end{cases}$$

nsc est alors une mesure de la complexité du problème. Si *nsc* est égal à zéro le problème est considéré comme facile et si *nsc* est négatif, plus cette valeur est faible plus le problème est difficile. C'est en fait une mesure déterminant à quel point un algorithme grimpeur utilisant le voisinage choisi aurait des chances de découvrir des points avec une valeur de la fonction *F* plus élevée. Quand la pente est négative, pour un *F(X)* donné, les valeurs que l'on trouve dans le voisinage de *X* sont en moyenne plus faibles que *F(X)*. Pour certains voisinages, $|\mathcal{V}o(X)|$ et l'ensemble des valeurs de *F(X)* distinctes peuvent être très grands. Cette approche est donc utilisée en calculant une approximation de nsc pour un échantillon des voisinages possibles d'un échantillon (biaisé vers les valeurs de *F(X)* supérieures) des points de *C*. La discrétisation par des ensembles d'abscisses et d'ordonnées peut diminuer fortement la capacité de cette mesure à représenter le degré de complexité d'un problème en "noyant" dans des moyennes les variations fines de la structure du paysage. Elle a également l'inconvénient d'être fortement dépendante du

voisinage choisi. Elle a cependant montré de bons résultats pour plusieurs problèmes classiques et pour un voisinage classique (mutation) de programmation génétique.

2.2.1.5 Les petits mondes

Le *petit monde* (small world) (Watts D.J., 1999) est une propriété topologique des graphes basée sur la longueur des chemins reliant les nœuds du graphe et sur le degré de "clusterisation" du graphe (Walsh T., 1999; Watts D.J. and Strogatz S.H., 1998). On appelle L la longueur moyenne des plus courts chemins reliant tous les couples de nœuds d'un graphe ayant n nœuds. Pour un nœud ayant k-1 voisins, il y a au plus $k(k-1)/2$ arcs entre eux (dans le cas maximum, les k nœuds forment une k-clique). Le coefficient de clusterisation d'un nœud est le rapport entre le nombre d'arcs existant effectivement avec ses k-1 voisins et le nombre d'arcs possibles. Le coefficient de clusterisation Cl d'un graphe est alors la moyenne de cette valeur pour tous les nœuds du graphe. On appelle L_{rand} et Cl_{rand} les valeurs de L et de Cl attendues pour des graphes aléatoires ayant le même nombre de nœuds et d'arcs. On définit le ratio de proximité μ d'un graphe par :

$$\mu = \frac{L \cdot Cl_{rand}}{Cl \cdot L_{rand}}$$

On considère que le graphe a une topologie de petit monde si $\mu >> 1$. Watts et Strogatz (Watts D.J. and Strogatz S.H., 1998) proposent une méthode de construction de graphe petit monde. On commence par construire un graphe régulier, c'est-à-dire un graphe où chaque nœud est connecté à ses k plus proches voisins. On choisit ensuite une probabilité p pour que chaque arc soit déplacé aléatoirement. Ainsi, si $p = 0$ le graphe est inchangé et donc régulier et si $p = 1$ le graphe est complètement aléatoire. Ils montrent ensuite que la propriété de petit monde apparaît pour un graphe, en fonction de n et de k, autour d'une valeur de p relativement précise et perd rapidement cette propriété quand la valeur de p augmente ou diminue à partir de cette valeur. Walsh (Walsh T., 1999) fait le lien entre cette propriété et la propriété bien connue de phénomène de seuil de complexité dans les problèmes classiques de coloriage de graphe, de planification ou les problèmes SAT (Cheeseman P. et al., 1991; Cook S.A. and Mitchell D.G., 1997; Gao Y., 2001; Monasson R. et al., 1999). Quand on construit des instances de ces différents problèmes de façon partiellement aléatoire en faisant varier cette proportion d'aléatoire, la complexité des problèmes va d'abord en augmentant avec cette proportion, puis il existe un seuil pour lequel le problème devient extrêmement difficile à résoudre, et une fois ce seuil dépassé la complexité des problèmes diminue de nouveau. L'hypothèse de Watts et Strogatz, pour expliquer ce phénomène lié à la notion de petit monde, est que les algorithmes de recherche utilisent souvent des stratégies basées sur des propriétés locales et sont donc trompés par des propriétés distantes incompatibles. La longueur moyenne des chemins L serait ainsi un indice de la quantité de propriétés distantes alors que le coefficient de clusterisation C serait un indice des propriétés locales. La valeur μ mesurant le rapport entre les deux représenterait le rapport entre ces propriétés distantes et locales et serait un indicateur de la complexité du problème. Pour valider cette hypothèse, Walsh (Walsh T., 1999) a étudié le comportement d'un algorithme de

coloriage de graphe, procédant par *retour arrière* (backtracking), sur toute une série d'instances de problèmes construites par la méthode donnée ci-dessus en faisant varier le paramètre *p*. Il a construit le graphe de la probabilité de devoir explorer plus de points de l'espace de recherche pour trouver le maximum en fonction du nombre de points déjà explorés. Il a montré que pour les valeurs de *p* correspondant à un graphe de type petit monde cette courbe suit une distribution "*à queue lourde*" (heavy-tailed). Ce type de distribution suit une loi de la forme :

$$\Pr(X > x) \approx c.x^{-\alpha}$$

avec c et α deux constantes positives. C'est donc une distribution à décroissance polynomiale. On peut l'interpréter comme suit : quand on augmente le nombre de points explorés on n'augmente que faiblement la probabilité de trouver la solution. Walsh montre ensuite qu'une stratégie d'exploration efficace est alors de lancer de multiples explorations successives avec départ aléatoire et de les arrêter très rapidement si elles ne trouvent pas la solution. Des travaux récents ont été menés pour comprendre l'apparition de ce phénomène heavy-tailed et définir une stratégie par retour arrière exploitant cette propriété et pour proposer des applications pour les problèmes SAT (Chen H. et al., 2001; Williams R. et al., 2003).

Le parallèle entre le concept de petit monde et celui de l'épistasie est évident. En effet, le degré d'épistasie correspond directement au paramètre "nombre de voisins" du petit monde et le graphe considéré est le graphe de dépendances défini dans la section 2.2.1.2. On retrouve également la notion de blocs contigus de la section 2.2.1.3 puisque la contiguïté est équivalente à la régularité du graphe de dépendance. On dispose ainsi d'une mesure, le ratio de proximité μ, qui prend en considération à la fois le degré d'épistasie, le chevauchement et la contiguïté du problème. Elle est donc *a priori* très attrayante. A ma connaissance elle n'a toutefois pas été utilisée dans ce contexte et il serait important de vérifier qu'elle prend bien en compte ces différentes propriétés de façon pertinente, c'est-à-dire en étant capable de mettre à jour les difficultés du problème provenant de plusieurs de ces propriétés en même temps. Il faudrait de plus comparer sa capacité à mesurer la complexité de chacune de ces propriétés indépendamment des autres par rapport aux mesures que l'on a présentées précédemment. Ce n'est de toute façon pas une mesure omnipotente puisqu'elle nécessite d'avoir une connaissance extrêmement précise de la structure du problème : le réseau complet d'interaction et d'ordonnancement des variables.

2.2.1.6 Discussion

Il existe de nombreuses définitions de la complexité des problèmes d'optimisation combinatoire. A ma connaissance il n'existe pas de publication les présentant toutes ensemble, en faisant une analyse comparative et en mettant à jour leurs points communs. C'est ce que j'ai essayé de faire dans cette section en les plaçant dans le cadre des paysages et des dépendances entre variables. Je présente également plusieurs mesures proposant de calculer le niveau de complexité d'un problème. Il apparaît qu'aucune de

ces mesures ne permet de prendre en considération toutes les propriétés impliquées dans la complexité. Plusieurs d'entre elles ne sont pas directement applicables à des problèmes de taille ne serait-ce que modeste du fait de leur complexité propre, souvent exponentielle avec la taille du problème. De plus, elles reposent pour la plupart sur un paysage particulier et donc sur un voisinage particulier. Or, nous l'avons vu, la complexité d'un problème peut varier considérablement suivant le voisinage choisi. Il serait utile de concevoir une mesure permettant de prendre en compte toutes ces propriétés en étant le plus indépendant possible d'un voisinage particulier. Une comparaison pour différents voisinages pourrait être une approche intéressante. Cela revient en fait à comparer différentes stratégies d'exploration de l'espace de recherche pour un même problème, la rapidité de la stratégie la plus efficace étant alors une mesure de la complexité du problème à résoudre. Il faut alors s'intéresser aux méthodes existantes pour résoudre ces types de problèmes et étudier leur comportement pour chaque propriété intervenant dans la complexité. Il serait alors essentiel de définir un problème paramétrable, peut-être une extension du NK-landscape, pouvant servir de base de comparaison entre ces différents algorithmes d'optimisation.

La section suivante porte sur une étude approfondie de l'efficacité des techniques d'exploration de type évolutionniste pour les différentes classes de complexité. Elle comporte également une présentation de plusieurs travaux préliminaires qui ont été menés par Olivier Martin, Julien Frey et moi-même.

2.2.2 Algorithmes évolutifs

La complexité des problèmes d'optimisation combinatoire est une notion qu'il est difficile de définir et de mesurer en raison du grand nombre de propriétés pouvant intervenir et la quantité de calculs nécessaire à leur évaluation. Le voisinage, qui produit un paysage particulier pour une fonction, est une composante majeure de la complexité. Cela implique qu'il existe une forte corrélation entre l'algorithme de recherche utilisé (et donc un ou plusieurs voisinages) et la fonction à optimiser : un algorithme particulier sera performant pour une classe de fonction donnée et réciproquement une fonction aura une complexité moindre si on découvre l'algorithme tirant efficacement parti de ses propriétés intrinsèques. Nous nous intéresserons dans cette section à l'étude de ce lien en considérant les méthodes d'explorations utilisées en recherche opérationnelle et celles appelées métaheuristiques (Michalewicz Z. and Fogel D., 2000; Yagiura M. and Ibaraki T., 2001) par la communauté de l'intelligence artificielle. Nous regarderons plus en détail une classe spécifique d'algorithmes dits évolutionnistes.

2.2.2.1 Considérations générales

2.2.2.1.1 Approches par construction

Il existe de très nombreuses stratégies d'exploration d'espace de recherche pour l'optimisation combinatoire. Certaines d'entre elles sont dites exactes, dans le sens où elles garantissent de découvrir le maximum de la fonction à optimiser. La plus simple

d'entre elles est l'exploration exhaustive de l'espace de recherche par arborescence, comme le font les algorithmes tels que le retour arrière par exemple (Russel S. and Norvig P., 2003). Leur complexité moyenne est exponentielle avec le nombre de variables. Des variantes plus ou moins complexes, comme les algorithmes *A** ou de *branch and bound*, utilisent des heuristiques exactes[6] pour élaguer l'espace de recherche. La complexité de ces méthodes est alors directement dépendante de l'heuristique utilisée et de son adéquation avec la fonction à optimiser. Le problème revient alors à choisir ou à découvrir la bonne heuristique, ce qui est la même chose que de découvrir le bon algorithme et est donc soumis aux mêmes contraintes de complexité que celles présentées dans la section précédente. Dans les cas difficiles ou si la fonction heuristique n'est pas adaptée, la complexité de la recherche reste exponentielle. Des extensions de ces approches, adaptées aux problèmes d'optimisation combinatoire sous contrainte, comme la *resolution search* (Chvatal V., 1997; Demassey S. et al., 2004) ou le *dynamic branch and bound* (Glover F. and Tangedahl L., 1976; Hanafi S. and Glover F., 2002) intègrent partiellement le concept d'épistasie. Leur principe est similaire et consiste en la découverte de combinaisons d'instanciations incompatibles du point de vue des contraintes du problème et en l'élagage de la partie correspondante de l'espace de recherche. Elles ne permettent toutefois que de découvrir des dépendances entre variables dues aux contraintes et ne s'appliquent donc pas à l'optimisation combinatoire globale. Elles ne s'appliquent de plus qu'à des problèmes linéaires. La *programmation dynamique* est aussi une approche exacte. Elle nécessite cependant que le problème soit fortement décomposable et, étant une approche gloutonne, qu'il ait de bonnes propriétés de contiguïté (les variables doivent être triées pour rassembler celles dépendant les unes des autres) pour garantir l'exactitude.

La plupart des autres méthodes ne garantissent de trouver la (ou les) solution(s) optimale(s) que dans certaines conditions liées aux propriétés de la complexité définies précédemment. Elles sont alors appelées non-exactes. La stratégie non-exacte sans doute la plus simple est la stratégie gloutonne dans laquelle on instancie les variables les unes après les autres, guidées par une heuristique déterminant l'ordre dans lequel choisir les variables et par la valeur potentielle de la solution non complètement instanciée. Un choix d'instanciation de variable ne peut plus être remis en cause une fois effectué. Cette stratégie a l'avantage d'être en général très rapide mais ne garantit en aucun cas de découvrir la solution optimale, les propriétés d'épistasie, trompeuse et de contiguïté des variables, n'étant pas prises en considération.

2.2.2.1.2 Approches par voisinage

Les méthodes que nous venons de voir sont dites également *par construction* dans le sens où elles construisent les solutions progressivement, variable après variable. Elles doivent donc disposer d'une heuristique estimant la qualité de la solution finale à partir de la solution partielle courante. Une large part des approches d'exploration non-exacte fonctionne au contraire avec une approche *par voisinage*. Cela signifie qu'elles disposent en permanence d'une ou plusieurs solutions complètes pour lesquelles on peut connaître

[6] Toujours au sens de garantissant la découverte de la solution optimale.

la valeur de F et qu'elles les modifient, plus ou moins drastiquement, pour obtenir de nouvelles solutions dans le voisinage des solutions précédentes.

Lorsque les problèmes sont linéaires (et bien sûr à valeurs entières), les méthodes de type *simplex* ou *primal-dual* peuvent souvent être très efficaces (Dantzig G.B., 1963; Papadimitriou C.H. and Steiglitz K., 1998). Etant basées sur des techniques d'ascension de gradient, elles nécessitent cependant la connaissance complète de la fonction à optimiser de façon à pouvoir calculer le gradient, ce qui n'est pas forcement le cas surtout pour des problèmes réels complexes. Elles sont de plus mal adaptées aux problèmes multinomiaux en raison du fait que les déplacements dans le paysage suivent un processus ascendant. Dans le cas général elles sont NP-complètes. Des variantes utilisant également le principe d'ascension de gradient, par exemple la programmation quadratique, existent pour les problèmes non linéaires. Fonctionnant selon les mêmes principes que les méthodes linéaires, elles sont soumises aux mêmes limitations. Ces méthodes sont souvent utilisées en combinaison avec d'autres algorithmes d'exploration tel que le branch and bound, jouant le rôle d'heuristiques d'élagage de l'espace de recherche.

Parmi les méthodes par voisinage, il existe une classe de méthodes appelées de *recherche locale* (local search), dont le fonctionnement général est le suivant : à partir d'un point de l'espace de recherche X_1, on choisit dans $\mathcal{V}o(X_1)$ un nouveau point X_2 en fonction des valeurs de $F(X_1)$ et de $F(X') \; \forall \; X' \in \mathcal{V}o(X_1)$, puis X_2 devient X_1 et on répète le processus jusqu'à une condition d'arrêt prédéfinie. Les grimpeurs stricts font bien évidemment partie de cette classe. Il en existe plusieurs variantes parmi lesquelles on peut citer : (1) *l'ascendant quelconque* pour laquelle X_2 est le premier élément de $\mathcal{V}o(X_1)$ calculé tel que $F(X_2) \geq F(X_1)$; (2) *le plus petit ascendant* pour laquelle X_2 est le plus petit élément de $\mathcal{V}o(X_1)$ tel que $F(X_2) \geq F(X_1)$; (3) *le plus grand ascendant* pour laquelle X_2 est le plus grand élément de $\mathcal{V}o(X_1)$ tel que $F(X_2) \geq F(X_1)$. Jones a mené une série d'expérimentations et d'études des bassins d'attractions de ces méthodes sur plusieurs problèmes, dont le NK-landscape, montrant une plus forte probabilité de trouver la solution pour la méthode (3) mais qu'également, reposant sur le calcul des voisinages complets (pour trouver le maximum), elle nécessite une quantité de calculs plus importante que la méthode (1) à résultats équivalents (Jones T., 1995). Une variante plus sophistiquée des grimpeurs stricts qui a suscité une importante littérature est la *recherche taboue* (tabu search) (Barnes J.W. et al., 1995; Battiti R. and Tecchiolli G., 1994; Glover F., 1986).

La base de cette approche est la même que celle des grimpeurs stricts, à la différence près qu'une liste, dite taboue, des dernières variables ayant été modifiées par les étapes précédentes de déplacement dans l'espace de recherche, est mémorisée. Elle est utilisée pour interdire de modifier les variables correspondantes pour le prochain mouvement à effectuer. Elle restreint donc le voisinage du point courant à ceux qui ne sont différents du point courant que pour des variables non-membres de la liste taboue. Cette liste est de taille limitée et les événements situés plus loin dans le passé que la taille de la liste sont oubliés. Il existe de nombreuses extensions à cette méthode comme la possibilité de faire varier la taille de la liste au cours de la recherche ou comme

l'aspiration qui permet de lever le caractère tabou d'un point du voisinage si celui-ci possède la plus haute valeur de la fonction objectif découverte jusqu'à présent. La recherche taboue permet de prendre en compte partiellement le concept d'épistasie par l'intermédiaire de la liste taboue regroupant plusieurs variables. Cela reste cependant très limité du fait que la sélection de ces variables se fait selon le principe du grimpeur. L'apport de la liste taboue ne permet donc pas d'échapper aux difficultés dues aux problèmes trompeurs ou aux problèmes fortement multinomiaux. La taille de la liste produit également un phénomène d'horizon qui peut s'avérer préjudiciable. De manière générale les grimpeurs stricts ne peuvent pas échapper aux pièges des bassins d'attractions des maximums locaux. La stratégie couramment suivie pour pallier ce problème est de faire des grimpeurs stricts à départs multiples. Le principe est de lancer la procédure d'exploration de nombreuses fois indépendamment à partir de points de départs aléatoires et de poursuivre les explorations à chaque fois jusqu'à l'obtention d'un maximum. Cette stratégie s'avère souvent efficace, nous en discuterons d'ailleurs plus amplement dans la section suivante. Le concept de petit monde tend en tout cas à démontrer que sur certains types de problèmes difficiles c'est la meilleure stratégie à suivre, cela du fait de la distribution à queue lourde de la probabilité d'atteindre le maximum global en fonction du nombre d'étapes d'exploration.

Les grimpeurs stochastiques font également partie de cette classe. Il en existe de nombreuses variantes et nous ne citerons ici que la plus classique : le *recuit simulé* (simulated annealing) (Kirkpatrick S. et al., 1983). Il existe dans la majorité des cas des versions stochastiques des grimpeurs stricts, en particulier pour la recherche taboue (Glover F. and Laguna M., 1997). Son principe est similaire à celui des grimpeurs stricts, la différence majeure étant que le choix du point X_2 est fait de manière probabiliste, avec la possibilité que $F(X_2) < F(X_1)$. La probabilité de sélection d'un point X' suit une loi décroissante avec un paramètre t égal au nombre d'étapes de recherche déjà effectuées. De façon générale, on peut définir cette probabilité par :

$$P(X') = \frac{1}{1 + e^{\frac{F(X_1) - F(X')}{g(t)}}}$$

Avec $g(t)$ une fonction décroissante sur t et t étant appelé température. Ainsi la probabilité de choisir un point X' diminue avec la différence $F(X_1)-F(X')$, c'est-à-dire que plus un point fait augmenter la valeur courante de $F(X)$ et plus il a de chance d'être sélectionné. De plus, cette probabilité augmente avec le temps (le nombre d'étapes de déplacement dans l'espace de recherche). Le comportement global du recuit simulé varie donc d'une stratégie proche du déplacement aléatoire en début d'exploration pour se transformer progressivement en une stratégie de type plus grand ascendant. Ce comportement stochastique permet au recuit simulé d'échapper aux pièges des bassins d'attractions des maximums locaux. Comme les autres méthodes de recherche locale, le voisinage utilisé est le voisinage de Hamming. L'efficacité de cette approche est donc très dépendante de la structure du problème. Mühlenbein a montré en 1992 (Muhlenbein H., 1992) que pour des problèmes décomposables en sous-problèmes indépendants d'ordre k, le nombre d'évaluations d'un grimpeur stochastique pour atteindre le

maximum peut être borné par $O(n^k \cdot \ln(n))$. Si *k* est fixé indépendamment de *n* et est très petit, les grimpeurs stochastiques peuvent résoudre efficacement ce type de problèmes polynomiaux en *n*. Mais si *k* dépend de *n*, le nombre d'évaluations n'est plus polynomial et même dans le cas où il est indépendant le problème ne peut plus être résolu en un temps raisonnable dès que la valeur de *k* est un peu plus élevée.

2.2.2.1.3 L'intelligence en essaim

Les différentes approches présentées jusqu'ici procèdent par une exploration séquentielle de l'espace de recherche. Il existe d'autres types de métaheuristiques basées sur une exploration parallèle avec coopération et/ou compétition. L'*intelligence en essaim* (swarm intelligence) est l'une de ces approches. Le principe des méthodes d'intelligence en essaim repose sur le mode de fonctionnement des insectes sociaux. Parmi celles-ci, les métaheuristiques d'optimisation par colonie de fourmis (ant colony optimization) sont certainement les plus étudiées (Bonabeau E. et al., 1999). Elles regroupent plusieurs algorithmes construits à partir d'un prototype initial appelé *système de fourmi* (ant system) proposé par Colorni en 1991 (Colorni A. et al., 1991). Ces algorithmes sont définis comme des systèmes multi-agents inspirés du comportement réel de colonies de fourmis. Leur principe est d'explorer simultanément l'espace de recherche par une population d'agents fourmis. Chaque fourmi construit une solution en réalisant des choix pour l'instanciation successive des variables du problème. Ces choix sont guidés par une communication indirecte entre les fourmis par l'intermédiaire d'une modification de leur environnement : elles déposent une quantité donnée de phéromones accessible localement par les autres fourmis et affectant leur comportement (et donc leurs choix). La quantité de phéromones déposée par une fourmi est proportionnelle à la qualité globale de la solution qu'elle a trouvée et est déposée sur tous les arcs parcourus par la fourmi (le chemin de chaque fourmi est mémorisé). Ce mécanisme est censé permettre l'apparition d'un comportement émergeant complexe de l'ensemble de la colonie (stigmergie). Ces algorithmes ont été appliqués avec succès sur une large classe de problèmes combinatoires difficiles comme le routage de véhicules, l'affectation de tâches ou le coloriage de graphe (Dorigo M. et al., 1999).

Du point de vue de l'optimisation combinatoire, ces algorithmes sont des algorithmes d'exploration par construction, réalisant des choix probabilistes de l'instanciation des variables biaisés par une heuristique locale (par utilisation d'une connaissance spécifique du problème) et par une mémoire globale portée par les phéromones. On peut également remarquer que les phéromones vont marquer les couples d'instanciations présents dans de bonnes solutions. C'est donc une méthode permettant de mettre à jour des dépendances entre variables et donc de résoudre les problèmes liés à l'épistasie. A ma connaissance, ces algorithmes n'ont pas été jusqu'à maintenant présentés et étudiés dans cette optique et il serait intéressant de les comparer avec les algorithmes cherchant à découvrir les dépendances que nous verrons dans la section suivante. On peut toutefois émettre quelques hypothèses : l'utilisation de l'heuristique locale complique fortement l'application de ces algorithmes à une large classe de problèmes et la comparaison avec d'autres algorithmes n'utilisant pas cette information;

le dépôt de phéromones et donc la potentielle détection de dépendances ne se fait qu'entre deux instanciations de couple de variables. Donc si le degré d'épistasie est supérieur à deux, les chances de découvrir les bonnes dépendances sont quasiment nulles en particulier pour des problèmes trompeurs (voir section 2.3.3.3).

2.2.2.2 Approche par exploration par échantillonnage

Une autre catégorie de métaheuristiques procède par exploration par échantillonnage, il s'agit de la catégorie des algorithmes évolutionnistes. Ils fonctionnent en mode "boite noire" c'est-à-dire sans connaître la fonction à optimiser[7]. La seule information dont ils disposent est la valeur de la fonction objectif pour les points considérés. Il existe deux manières de concevoir ce genre de stratégies. La première, que l'on peut qualifier de classique, consiste à dire que l'on dispose d'une population de solutions (appelées *chromosomes*) en compétition/coopération parmi laquelle on sélectionne les meilleurs (du point de vue de la valeur de la fonction objectif). On applique sur les chromosomes sélectionnés une série d'opérateurs "génétiques" les transformant en une nouvelle population de chromosomes. On répète le processus jusqu'à l'obtention d'une condition d'arrêt. La deuxième, que l'on peut qualifier de statistique, fonctionne de façon similaire. La population est toutefois considérée comme un échantillon de l'espace de recherche sur lequel on réalise une sélection biaisée vers les meilleurs. On construit ensuite un modèle statistique de cet échantillon biaisé et on utilise le modèle pour générer un nouvel échantillon. Ce sont ces deux approches que nous allons décrire dans les deux prochaines sections.

2.2.2.2.1 L'approche classique

Le concept d'algorithme évolutionniste a près de quarante années d'existence. Il a depuis été décliné sous de nombreuses formes dont les plus connues sont : la programmation évolutionniste (Fogel L.J. et al., 1966), les algorithmes génétiques (Holland J.H., 1975), les stratégies évolutives (Rechenberg I., 1973) et la programmation génétique (Koza J.R., 1992). Nous nous intéresserons ici plus en détail aux algorithmes génétiques qui ont été popularisés par le fameux livre de 1989 de Goldberg (Goldberg D.E., 1989a). Des versions permettant l'optimisation de problèmes à variables réelles (Michalewicz Z., 1996; Tsutsui S. et al., 1999) ou de problèmes avec contraintes (Eiben A.E., 2001) existent mais nous ne nous y attarderons pas ici.

L'algorithme génétique standard nécessite une étape d'initialisation au cours de laquelle une population, de taille fixée *Tp* de vecteurs *X* (les chromosomes), est générée aléatoirement. Chaque vecteur correspond donc à un point de l'espace de recherche et donc aussi à une instanciation donnée de chacune des variables. L'algorithme consiste en une répétition d'une succession de trois étapes : (1) on associe à chaque vecteur *X* une valeur de *fitness* égale à *F(X)*; (2) on sélectionne avec remise *Tp* vecteurs dans cette population avec une probabilité dépendant de la valeur de fitness de chaque vecteur; (3)

[7] Contrairement aux approches par gradient mais comme les autres métaheuristiques en général.

on applique de manière stochastique une série d'opérateurs génétiques sur ces vecteurs et les vecteurs ainsi transformés sont placés dans une nouvelle population. C'est à cette nouvelle population, de taille *Tp* également, que sont réappliquées les trois étapes (appelées *génération*). Il existe de très nombreux opérateurs de sélection, les plus courants étant la sélection proportionnelle à la fitness, la sélection élitiste et la sélection par tournois. Elles permettent de moduler la pression de sélection exercée sur la population, c'est-à-dire l'importance du biais statistique vers les meilleures solutions. Les opérateurs génétiques de l'étape (2) sont également très divers, les plus classiques étant la *mutation* ponctuelle d'un gène (une variable dont l'instanciation est changée), et les *crossovers* uniformes ou à un ou deux points dans lesquels l'instanciation d'une succession de variables est échangée entre deux vecteurs. Ce sont ces opérateurs qui permettent l'exploration de l'espace de recherche en transformant un point en un autre. Comme nous l'avons vu, à chaque opérateur *o* correspond une fonction de voisinage $\mathcal{V}o$ qui elle-même définit un paysage particulier (Jones T., 1995). Un grand nombre de paramètres est nécessaire pour les algorithmes génétiques, par exemple la taille de la population, les différentes probabilités associées aux opérateurs, la pression de sélection… De nombreuses études expérimentales ont été menées pour évaluer l'influence de ces différents paramètres, les conclusions étant que, en général, cela dépendait fortement des problèmes considérés. D'autres études ont également été réalisées pour comparer les capacités d'exploration des algorithmes génétiques et plus particulièrement des opérateurs de crossover avec des approches de grimpeurs stricts à départs multiples (Baluja S., 1996; Goldberg D.E. and Segrest P., 1987; Jones T., 1995; Sharpe O.J., 2000). Les résultats semblent montrer que dans la grande majorité des cas, et plus particulièrement sur des problèmes trompeurs, pourtant spécifiquement conçus pour être difficiles pour les approches de type grimpeur, les algorithmes génétiques standard utilisant les opérateurs classiques ne font quasiment pas mieux et même souvent moins bien que les grimpeurs. Nous présentons toutefois des résultats plus complets dans la section 2.3.3.1 qui mettent en évidence dans quelles circonstances les uns ou les autres fonctionnent mieux.

Plusieurs études théoriques ont été menées pour essayer de comprendre le comportement des algorithmes génétiques lors de la résolution d'un problème. Une première approche consiste à étudier le comportement exploratoire de l'algorithme génétique comme un processus markovien (Goldberg D.E. and Segrest P., 1987; Rowe J.E., 2001; Vose M.D. and Liepins G.E., 1991). Malheureusement la taille des matrices markoviennes à manipuler et la complexité de la modélisation des opérateurs limitent ces études aux versions les plus simples des algorithmes génétiques et les conclusions restent limitées à la démonstration de convergence de l'algorithme vers un point fixe. L'application à des problèmes de taille, même extrêmement raisonnable, d'une analyse par processus de Markov n'est pas envisageable car elle nécessite la construction de tables de probabilités de transitions de taille 2^{2*n*Tp}! Une publication récente (Moey C.J. and Rowe J.E., 2004) propose de contourner cette difficulté en regroupant les états du processus markovien selon les valeurs de fitness des différents vecteurs. Cela mène à des résultats prometteurs sur la capacité de réduction de la complexité de cette approche sans perte significative de précision. De manière à pouvoir étudier le comportement des algorithmes génétiques sur des problèmes de plus grande taille, Prügel-Bennett et Rogers

proposent une approche basée sur l'étude des cumulants (moyenne, écart type,…) de la valeur de fitness d'une population de chromosomes au cours des générations successives (Prugel-Bennett A. and Rogers A., 2001). Pour permettre l'évaluation de ces différentes valeurs, ils ont dû se restreindre à un problème très simple : le *comptage des uns* (aussi connu sous le nom de *onemax*). Ce problème consiste à rechercher le maximum de la fonction :

$$F(X) = \sum_{i=1}^{n} x_i$$

Ils supposent dans un premier temps que la population est de taille infinie et que les valeurs de fitness dans la population suivent une distribution normale. Ils donnent alors l'expression de la variation de la moyenne μ et la variance σ^2 de la fitness de la génération t à la génération *t + 1*, avec une sélection par tournoi de taille 2 et une mutation ponctuelle de probabilité *u*, sous la forme :

$$\mu(t+1) = un + (1-2n)\mu(t) + \frac{(1-2u)}{\sqrt{\pi}}\sigma(t)$$

et

$$\sigma^2(t+1) = u(1-u)n + (1-2u)^2(1-\frac{1}{\pi})\sigma^2(t)$$

Ils montrent une bonne correspondance entre les valeurs théoriques prédites et des résultats expérimentaux avec en particulier une rapide décroissance de la variance puis une stabilisation et une croissance plus lente de la moyenne jusqu'à un plateau. Ils étendent ensuite ce modèle au cumulant d'ordre 3, à une population de taille finie et à l'opérateur de crossover uniforme. Ils constatent également une bonne correspondance avec les résultats expérimentaux et le même genre de phénomènes pour la moyenne et la variance, le crossover ayant toutefois pour effet de ralentir la décroissance de la variance et d'augmenter par là-même la vitesse de croissance de la moyenne. Ces différents travaux se limitent pour l'instant à des problèmes simples et il paraît difficilement envisageable de les appliquer à des problèmes plus complexes faisant intervenir l'épistasie.

La première tentative d'analyse du comportement de convergence des algorithmes génétiques classiques faisant intervenir la notion d'épistasie est liée au concept de *motifs de similarité* (*schemes*) (Holland J.H., 1968). L'épistasie n'étant pas encore associée aux algorithmes évolutifs, le terme utilisé était celui de *brique élémentaire* (*building block*) lui-même introduit très rapidement dès les débuts de l'approche évolutionniste (Bagley J.D., 1967). Le principe des motifs de similarité est de représenter des généralisations de solutions grâce à l'extension de l'alphabet utilisé pour les variables x_i au triplet {0,1,*}. Le caractère * peut être interprété comme "0 ou 1" et représente donc en même temps les deux instanciations possibles d'une variable. Ainsi un vecteur *Xm* contenant des 0, des 1

et des * peut être vu comme un ensemble de vecteurs E_{Xm} ne contenant que des 0 et des 1. Par exemple, le vecteur *Xm* = 1*00* représente l'ensemble de vecteurs E_{xm} = {10000, 10001, 11000, 11001}. On dit également qu'un motif de similarité *Xm* est contenu dans un vecteur *X* si $\exists X' \in E_{Xm}$ tel que *X'* = *X*. Il existe donc 3^n motifs de similarité différents pour un vecteur de taille *n* et un vecteur *X* donné contient 2^n motifs de similarité différents. On appelle l'ordre de *Xm*, noté O(*Xm*) le nombre de variables non instanciées à *. On appelle la longueur utile de *Xm*, notée L(*Xm*), la distance (en positions dans le vecteur) entre la première et la dernière position du vecteur non instanciée à *. Par exemple L(110*0**) = 5 – 1 = 4. Goldberg (Goldberg D.E., 1989a) utilise ces notions pour essayer d'expliquer le fonctionnement des algorithmes génétiques classiques[8]. Remarquant que la probabilité qu'un motif *Xm* survive à un crossover à un point est de $p_s = 1 - L(Xm)/(n-1)$, il calcule le nombre *N(Xm,t+1)* de vecteurs de la population contenant *Xm* à la génération *t* + 1 sachant le nombre *N(Xm,t)* de vecteurs de la population contenant *Xm* à la génération *t* par :

$$N(Xm,t+1) = N(Xm,t) \cdot \frac{\overline{F}(Xm)}{\overline{F}} \cdot \left(1 - p_c \cdot \frac{L(Xm)}{n-1} - p_m \cdot O(Xm)\right)$$

avec $\overline{F}(Xm)$ la fitness moyenne des vecteurs contenant le motif *Xm* à la génération *t*, $\overline{F}$ la fitness moyenne de la population à la génération *t*, p_c la probabilité d'une mutation sur une variable et p_m la probabilité pour qu'il y ait un crossover entre deux vecteurs. Il apparaît donc qu'un motif dont la fitness est située au-dessus de la moyenne va être sélectionné et combiné préférentiellement par l'algorithme génétique. L'hypothèse de la brique élémentaire en découle. L'idée est ici que la fonction à optimiser est composée de sous-fonctions, chacune portant sur un sous-ensemble des variables représentées par un motif et appelé brique élémentaire. L'algorithme génétique va donc sélectionner et combiner les briques élémentaires les meilleures pour découvrir plus rapidement la bonne solution. Goldberg montre en fait que l'algorithme génétique manipule Tp^3 motifs en parallèle et que c'est cela qui explique son efficacité pour résoudre ce type de problème. On voit donc bien ici apparaître les prémices de l'épistasie et une tentative de preuve de l'efficacité des algorithmes génétiques pour traiter ce genre de problèmes décomposables.

Thierens et Goldberg ont réalisé plusieurs travaux pour évaluer la capacité des algorithmes génétiques classiques à résoudre des problèmes trompeurs (Sastry K. and Goldberg D.E., 2002; Thierens D., 1999; Thierens D. and Goldberg D.E., 1993; Thierens D. and Goldberg D.E., 1994). Ils ont en particulier évalué, dans le cas de fonctions composées d'une juxtaposition de sous-fonctions trompeuses de tailles *k*, le temps *t* (en nombre de générations) nécessaire pour que les briques élémentaires correspondant aux variables des sous-fonctions trompeuses soient "mixées" correctement pour produire la solution complète. Ce temps peut être défini par :

[8] Ici l'algorithme génétique utilisé consiste en une sélection proportionnelle à la fitness, une mutation ponctuelle et un crossover à un point.

$$t = c_1 \cdot \frac{2^{c_2 k}}{Tp \cdot p_c} \cdot \frac{2^m}{m^{\frac{5}{2}}}$$

avec c_1 et c_2 deux constantes, m le nombre de sous-fonctions de taille k et p_c la probabilité d'occurrence de l'opérateur de crossover (uniforme). Ils en déduisent une relation sur la taille de la population nécessaire pour la convergence de l'algorithme vers le maximum global :

$$Tp \cdot \ln Tp > c_1 \cdot \frac{2^{c_2 k} \cdot \ln s}{p_c} \cdot \frac{2^m}{m^{\frac{5}{2}}}$$

avec s la pression de sélection. Ces calculs montrent que le temps de convergence et la taille de la population de l'algorithme génétique avec crossover à un ou plusieurs points sont exponentiels avec la taille le nombre de dépendances et le nombre de sous-fonctions. Ils en concluent que ces algorithmes ne peuvent pas résoudre les problèmes trompeurs et qu'il faut concevoir de nouveaux algorithmes pour pouvoir les résoudre. Ce sont ces travaux qui ont été à l'origine du développement de l'approche évolutionniste cherchant à découvrir les dépendances entre variables pour résoudre ce type de problèmes que nous verrons dans la section suivante.

On peut cependant remarquer que les calculs réalisés ne prennent en considération que la sélection et le crossover et ne font pas intervenir la mutation. Nous montrons expérimentalement dans les sections 2.3.2 et 2.3.3 que l'utilisation conjointe de la mutation et du crossover à un et deux points permet de résoudre très efficacement certain types de problèmes difficiles. Nous étudions également dans la section 2.3.3 l'effet du crossover sur la survie des briques élémentaires et montrons qu'elle est fortement corrélée à la contiguïté de ses variables, c'est-à-dire à sa longueur utile, ce qui n'a pas été pris en compte dans les travaux précédents.

2.2.2.2.2 L'approche statistique

Parallèlement aux développements continus sur l'approche classique, une nouvelle façon de concevoir les algorithmes génétiques est apparue : les *algorithmes génétiques par construction de modèle probabiliste* (probabilistic model-building genetic algorithms, *PMBGA*) (Muhlenbein H. and Mahnig T., 2001; Pelikan M., 2002). Leur principe est de considérer la population de chromosomes de l'algorithme génétique comme un échantillon de l'espace de recherche, d'en construire un modèle probabiliste et d'utiliser ce modèle pour générer une nouvelle population. L'algorithme général est :

(1) $t = 0$
    générer la population initiale *P(t)*
(2) sélectionner un ensemble de solutions (chromosomes) prometteuses *S(t)*
(3) construire un modèle statistique *M* de cet ensemble
(4) générer un nouvel ensemble de solutions *O(t)* à partir de ce modèle
(5) créer une nouvelle population *P(t+1)* en remplaçant certaines solutions de *P(t)* par des solutions de *O(t)*
    $t = t + 1$
(6) si le critère de terminaison n'est pas atteint, aller en (2)

Les premiers travaux dans ce domaine ont été menés par Mühlenbein et Voigt en 1996 (Muhlenbein H. and Voigt H.M., 1996). L'idée était de transformer l'étape de recombinaison habituellement réalisée par des opérateurs de crossover à un ou plusieurs points par une recombinaison globale sur toute la population. Cette méthode est appelée *gene pool recombination.* Dans ce modèle de recombinaison, on mesure les fréquences $f(x_i = 0)$ et $f(x_i = 1)$ pour tout *i* et pour tout *X* de l'ensemble *S(t)* sélectionné. Le modèle *M* est alors un vecteur contenant toutes ces fréquences. En considérant les fréquences comme les probabilités $p(x_i = 0)$ et $p(x_i = 1)$ que chaque variable soit instanciée à 0 ou à 1 dans le maximum global, on génère la nouvelle population *O(t)* en utilisant ces probabilités. Cet algorithme, appelé *algorithme à distribution marginale univariée* (univariate marginal distribution algorithm, UMDA), est le premier à faire partie de la classe des PMBGAs. Harik et al. ont montré que cette approche converge vers le maximal global dans le problème onemax en $O(n \cdot \ln n)$ si la population a une taille appropriée (Harik G.R. et al., 1997). Cet algorithme extrait des statistiques à partir de la population pour chaque variable prise indépendamment. Cela a pour conséquence que ses performances deviennent très mauvaises (et en particulier inférieures à celles d'un grimpeur strict) dès que le degré d'épistasie du problème devient supérieur à zéro.

Pour contourner cette difficulté, d'autres versions de PMBGA ont été proposées prenant en compte des distributions de probabilités de plusieurs variables à la fois. L'*algorithme de distribution factorisée* (factorized distribution algorithm, FDA) (Muhlenbein H. et al., 1999) par exemple, propose de construire un modèle *M* de la dépendance entre variables par un réseau bayésien (Frez B.J., 1998; Jensen F.V., 2001). Un réseau bayésien est un modèle représentant la distribution de probabilité d'un vecteur *X* de variables x_i, à partir de probabilités conditionnelles entre ces variables :

$$p(X) = \prod_{i=1}^{n} p(x_i \mid \Pi_i)$$

avec Π_i l'ensemble des variables dont dépend la variable x_i. Il peut être représenté sous la forme d'un graphe dont les nœuds sont les variables et les arcs les dépendances entre les variables. La figure 1 présente un exemple d'un tel graphe pour des variables liées à la

probabilité d'avoir un accident de voiture. Cet algorithme s'avère très efficace pour peu que l'on lui fournisse un échantillon de taille suffisante (fonction du nombre *k* des dépendances comme nous le verrons plus loin) et que l'on connaisse déjà la décomposition du problème (c'est-à-dire quelles variables dépendent de quelles autres) car il faut lui fournir la structure du réseau. Cette efficacité n'est guère étonnante puisque, comme on l'a vu dans la section 2.2.1.2 quand un problème est décomposable sa complexité est considérablement réduite. Une grande partie de la difficulté provient justement de la découverte de ces dépendances avant de pouvoir les exploiter.

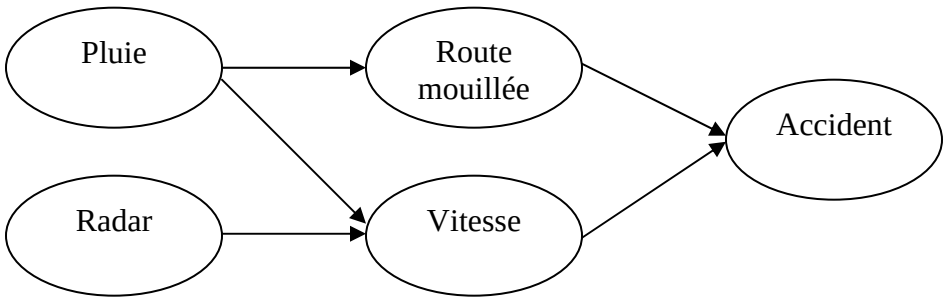


Figure 1. Exemple d'un réseau bayésien modélisant la probabilité d'un accident de voiture

L'approche appelée *découverte des dépendances* (linkage learning) a été imaginée pour traiter cette difficulté (Harik G.R., 1997). Nous nous intéresserons plus particulièrement à une sous-classe appelée *probabilistic linkage learning* (Pelikan M., 2002) ou *learning a Bayesian network* (LFDA) (Muhlenbein H. and Mahnig T., 2001). Le principe est d'apprendre un réseau bayésien représentant les dépendances entre les variables du problème en même temps que l'algorithme génétique fait son exploration de l'espace de recherche. C'est l'approche choisie par Pelikan pour son algorithme BOA (*Bayesian optimization Algorithm*) (Pelikan M. et al., 1999; Pelikan M., 2002). L'apprentissage d'un réseau bayésien nécessite deux choses : découvrir la structure du réseau, c'est-à-dire découvrir les dépendances, et déterminer les probabilités conditionnelles associées aux arcs. Déterminer les probabilités peut se faire très simplement en procédant de façon similaire à ce qui est fait dans l'algorithme UMDA. On estime chaque probabilité conditionnelle à partir de la fréquence observée dans l'échantillon (la population) des occurrences simultanées de chaque instance de chaque variable. L'apprentissage du réseau est un problème beaucoup plus complexe puisqu'il revient à déterminer quel réseau parmi les 2^{n^2} possibles[9] représente le mieux les données à modéliser (l'échantillon). Ce problème est NP-complet en soi (Chickering D.M. et al., 1997) et il faut utiliser une approche heuristique non exacte pour construire le réseau. Il n'est donc pas possible d'assurer la qualité du réseau obtenu. La méthode utilisée dans BOA pour construire le réseau est une méthode gloutonne qui va construire le réseau petit à petit à partir du réseau sans aucun arc. Il faut pour cela disposer d'une métrique qui

[9] Cela correspond à toutes les manières de placer les $O(n^2)$ arcs possibles entre tous les couples de variables.

mesure la qualité du réseau (à quel point le réseau est un bon modèle des données) à chaque étape de la construction. L'algorithme de construction est alors :

(1) initialiser le réseau *R* sans aucun arc
(2) choisir parmi les opérations de base (ajout, retrait ou retournement) pour tous les arcs possibles celle qui augmentera le plus la mesure de qualité du réseau.
(3) si le critère de terminaison n'est pas atteint aller en (2)

La métrique utilisée dans l'algorithme BOA est la métrique du *critère d'information bayésienne* (bayesian information criterion, BIC) (Schwarz G., 197). Elle est définie par :

$$BIC(R) = \sum_{i=1}^{n} \left(-H(x_i \mid \Pi_i) \cdot Tp - 2^{|\Pi_i|} \cdot \frac{\log_2(Tp)}{2} \right)$$

avec $H(x_i \mid \Pi_i)$ l'entropie conditionnelle de x_i sachant ses parents Π_i. Elle se définit par :

$$H(x_i \mid \Pi_i) = -\sum_{\pi_i} p(x_i = 0, \pi_i) \cdot \log_2(p(x_i = 0, \pi_i)) - \sum_{\pi_i} p(x_i = 1, \pi_i) \cdot \log_2(p(x_i = 1, \pi_i))$$

avec π_i l'une des instanciations possibles des variables parentes de x_i. Il n'existe pas de mesure parfaite de la qualité d'un réseau bayésien. La mesure BIC a tendance à être sensible au bruit dans les données et à surestimer le nombre de dépendances. Le processus d'apprentissage du réseau bayésien étant heuristique et basé sur une mesure susceptible de produire des erreurs, on ne peut garantir que le réseau final aura bien découvert toutes et seulement toutes les dépendances de l'échantillon. BOA étant un outil commercial, l'algorithme précis de construction du réseau n'a pas été publié.

L'analyse de complexité de l'algorithme faite dans (Pelikan M. et al., 2000) laisse certains points dans le flou. Le calcul de complexité est donné dans le cas où seules des opérations d'ajout d'arcs sont effectuées et dans le cas où la valeur de k est spécifiée à l'algorithme. Le calcul se décompose de la manière suivante. On peut évaluer l'apport à la mesure de qualité du réseau (ici la mesure *Bayesian Dirichlet Metric*, BD) de chaque arc quand le réseau est vide en $O(n^2 \cdot Tp)$. On remarque aussi que l'on peut choisir le meilleur arc à ajouter à chaque étape de l'algorithme glouton en $O(n^2)$. Recalculer la nouvelle mesure de qualité de réseau se fait alors en $O(2^k \cdot n \cdot Tp)$. Les étapes de choix de l'arc et de recalcul de la mesure doivent se faire $k \cdot n$ fois. La complexité totale est donc de $O(k \cdot 2^k \cdot n^2 \cdot Tp + k \cdot n^3)$. Ce calcul laisse sous-entendre que le choix de l'arc à ajouter à chaque étape ne se fait pas en calculant le gain en mesure pour le réseau total pour chaque arc possible, (on aurait alors une complexité en $O(k \cdot 2^k \cdot n^4 \cdot Tp + k \cdot n^3)$), mais sur le gain en mesure si le réseau était vide (valeurs pré-calculées en $O(n^2 \cdot Tp)$). Dans ce cas, les choix sont effectués uniquement en fonction des dépendances de couples

de variables et sont susceptibles d'être biaisés par des fonctions trompeuses pour lesquelles les bonnes dépendances n'apparaissent que lorsque l'on observe les k + 1 variables à la fois.

L'étape suivante de l'approche PMBGA est de générer la nouvelle population à partir du modèle construit sur la population courante. Dans BOA, c'est donc le réseau bayésien qui est utilisé pour générer la nouvelle population. Le pseudo code donné dans (Pelikan M. et al., 2000) n'est guère plus clair que le calcul de complexité précédant :

(1) marquer toutes les variables comme non instanciées
(2) prendre une variable x_i marquée non instanciée dont tous les parents Π_i sont déjà instanciés à π_i
(3) instancier x_i à 0 avec la probabilité $P(x_i = 0 \mid \Pi_i = \pi_i)$ et à 1 sinon
(4) marquer x_i comme instanciée
(5) s'il reste des variables non instanciées retourner en (2)

Comment l'étape (2) peut-elle commencer si toutes les variables sont marquées comme non instanciées? Il doit donc y avoir une étape préalable qui instancie aléatoirement une série de variables mais cela n'est pas indiqué. On peut se poser la question de savoir quelles implications cela peut avoir sur la qualité des solutions générées si une partie de la génération se fait en dehors du modèle. La complexité totale de la génération d'une nouvelle population est donc en $O(k \cdot n \cdot Tp)$.

Pelikan et Goldberg (Goldberg D.E., 2002; Pelikan M., 2002) proposent une preuve de la convergence de la méthode BOA basée sur le calcul de la taille de la population nécessaire à cette convergence. Une partie de ces travaux est basée sur ceux déjà effectués pour l'algorithme génétique classique (Goldberg D.E., 1989b; Goldberg D.E. et al., 2001; Holland J.H., 1975). Ils reposent sur la constatation suivante. Le nombre attendu de copies m(B_i) dans une population de taille Tp de chaque brique élémentaire B_i de taille k est de :

$$m(B_i) = \frac{Tp}{2^k}$$

Ce qui signifie que pour assurer la présence d'un nombre donné de copies de chaque brique élémentaire dans la population, sa taille doit croître exponentiellement avec la taille k des blocs (donc le degré d'épistasie). Une analyse plus détaillée de la taille de la population, pour que, avec une probabilité égale à 1/m, toutes les instanciations possibles de tous les blocs de taille k soient présentes au moins une fois dans la population initiale, conduit au résultat suivant :

$$Tp \approx 2^k (2 \cdot \ln(m) + k \ln(2))$$

avec m le nombre de blocs de taille k dans la fonction de taille n. Dans le cadre de l'approche BOA, plusieurs autres facteurs interviennent dans le calcul de la taille de la

population. En premier lieu, il y a le fait que la sélection doive faire le bon choix parmi les différentes instanciations d'un même bloc qui conduit à une taille de la population en $O(2^k \cdot \sqrt{n})$. Vient ensuite le problème de la *dérive génétique* (genetic drift), c'est-à-dire le fait que la meilleure instanciation d'un bloc puisse être perdue avant la convergence globale vers la bonne solution. Ce facteur conduit à une taille de population en $O(n)$. Enfin il y a le problème de la construction du modèle statistique de la population. Dans ce cas, on doit garantir que toutes les dépendances soient effectivement détectées et que les fréquences associées soient les bonnes. Cela nécessite une population de taille en $O(2^k \cdot n^{1.05})$. Ce dernier terme étant prépondérant, on en déduit que globalement la taille de la population est en $O(2^k \cdot n^{1.05})$. Ces calculs sont toutefois effectués avec un certain nombre d'approximations. Il est supposé que la valeur $F_i(X)$[10] a une distribution normale dans la population. Le modèle statistique est supposé parfait, c'est-à-dire supposé modéliser parfaitement la population, alors qu'il est en fait construit par une approche heuristique assez brutale (gloutonne). La mesure de qualité du réseau bayésien est la mesure BIC, alors que dans le calcul de complexité de BOA, c'est la mesure BD qui est utilisée. Enfin, et cela est vrai aussi pour les calculs réalisés sur l'algorithme génétique classique, l'opérateur de mutation n'est jamais pris en compte[11]. Or, la mutation est clairement un apport de diversité pour la population et doit avoir des conséquences importantes sur la taille de la population nécessaire pour que toutes les instanciations de tous les blocs soient rencontrées en cours d'exploration. On peut donc supposer que les valeurs calculées surestiment cette taille lorsqu'un opérateur de mutation est présent. Quoi qu'il en soit, à partir de ces calculs, Pelikan et Goldberg donnent la preuve de convergence de leur algorithme en $O(\sqrt{n})$ générations. En reprenant les valeurs annoncées sur la complexité de construction du réseau bayésien, on en déduit la complexité totale de la méthode BOA en $O(k \cdot n^{7/2} \cdot (2^{2k} + 1))$. Ces calculs sont valables dans le cas où le problème considéré possède des blocs de taille *k* complètement indépendants les uns des autres.

Pelikan et Goldberg ont proposé une extension à l'approche BOA appelée *algorithme hiérarchique d'optimisation bayésienne* (hiararchical bayesian optimization algorithm, hBOA) (Pelikan M. et al., 2000; Pelikan M. and Goldberg D.E., 2001). Son but est de résoudre des problèmes trompeurs à plusieurs niveaux, c'est-à-dire des problèmes dans lesquels il existe des blocs de blocs, chaque niveau de blocs ayant sa propre sous-fonction de fitness. Le principe de hBOA est similaire à celui de BOA à ceci près que hBOA utilise des graphes de décisions (Jensen F.V., 2001) pour stocker les tables de fréquences des dépendances entre variables et un système de niches écologiques au sein de la population. Les graphes de décisions permettent un stockage beaucoup plus compact de ces fréquences que les tables utilisées dans BOA et ont donc besoin de moins de calcul pour être construits. Les niches écologiques permettent la préservation de blocs

[10] Qui est la valeur de la sous-fonction correspondant au bloc de variables *i*.
[11] L'opérateur de mutation n'existe pas explicitement dans hBOA mais il a son équivalent. Le processus de génération de solutions à partir du modèle est tout à fait à même de générer des instanciations de blocs non présentes dans la population initiale.

sous-optimaux de façon à ce qu'ils puissent être combinés pour former des blocs de plus hauts niveaux.

L'approche hBOA a été validée sur deux problèmes difficiles les *verres de spins* (spin glasses) et le problème MAXSAT (Pelikan M. and Goldberg D.E., 2003). Pour le premier problème, les résultats montrent que hBOA a besoin de $O(n^{4.25})$ évaluations pour découvrir le maximum alors que la meilleure méthode existante le découvre en $O(n^{3.5})$. De plus, ce calcul ne prend pas en compte la complexité de la construction du réseau bayésien pour l'approche hBOA. On peut, par contre, constater, que contrairement à l'autre méthode, hBOA n'utilise aucune information spécifique au problème de verres de spins pour le résoudre. D'autres résultats montrent que hBOA couplé avec un grimpeur strict a cette fois besoin de moins de temps de calcul que la meilleure méthode pour ce problème. Pour le problème MAXSAT, hBOA a été couplé avec un algorithme classique utilisé sur ce problème : GSAT. Cet algorithme est aussi un grimpeur strict. Sur la majeure partie des instances du problème MAXSAT, hBOA+GSAT fait moins bien que le grimpeur stochastique WalkSAT. hBOA+GSAT parvient toutefois à résoudre certains problèmes insolubles pour WalkSAT.

Les derniers travaux sur l'approche par construction de modèles apportent un certain nombre d'améliorations à hBOA et à d'autres algorithmes de PMBGA (Sastry K. et al., 2004). Un premier travail porte sur le concept de *mutation Building-Block-Wise* (Sastry K. and Goldberg D.E., 2004b; Sastry K. and Goldberg D.E., 2004a). Son principe consiste à découvrir les briques élémentaires à partir du modèle statistique de la population puis d'utiliser une mutation fonctionnant comme un grimpeur strict sur le sous-espace des instanciations possibles de chaque brique. Ainsi, pour des briques élémentaires de taille *k*, 2^k évaluations de fonction de fitness sont nécessaires pour découvrir le maximum de chaque brique, donc au total $m \cdot 2^k$ évaluations si le problème est composé de m briques élémentaires. Cette approche revient en fait à décomposer le problème en ses sous-problèmes indépendants puis à résoudre chacun d'eux séparément, ce qui est bien sûr beaucoup plus rapide. Il faut pour cela toujours garantir une taille de population permettant de découvrir le bon modèle et que le problème soit décomposable en sous-problèmes indépendants. Un autre travail porte sur la notion *d'héritage de fitness* (fitness inheritance) (Pelikan M. and Sastry K., 2004). L'idée est ici d'éviter de calculer explicitement la fitness de chaque nouveau point exploré de l'espace de recherche en en faisant une estimation. Cette estimation est faite en héritant de l'information de la population précédente dans laquelle un certain nombre de points ont été effectivement évalués. On utilise le modèle statistique pour déterminer quelles sont les briques élémentaires, puis, pour chacune d'elles, on estime leur contribution à la fitness totale en étudiant la distribution de la fitness des solutions qui les contenait dans la population précédente. Pour une fraction de la nouvelle population, ce sont ces valeurs estimées qui sont utilisées pour calculer leur fitness. Malgré la perte de précision liée à cette estimation, le gain global en nombre de points effectivement évalués peut être important.

Une étude récente sur l'efficacité de la parallélisation d'algorithme basée sur la découverte des dépendances (Munetomo M. et al., 2004) propose une comparaison intéressante sur l'efficacité de l'approche hBOA et celle d'une autre approche

importante : *l'identification de dépendance par contrôle de non-linéarité* (linkage identification by nonlinearity check, LINC). Comme nous l'avons évoqué dans la section 2.2.1.2, la notion de dépendance entre variables peut être interprétée comme une non-linéarité dans la contribution à la fitness totale de ces variables. C'est ce constat qui est à la base de LINC. Le principe de LINC est de générer une population échantillon de taille suffisante (en $O(c2^k)$ avec c dépendant de la probabilité de se tromper en détectant une non-linéarité). Puis, pour chaque vecteur X de la population, on calcule pour tous les couples de variables x_i et x_j les valeurs suivantes :

$$\Delta F_i(X) = F(..\overline{x_i}.....) - F(..x_i.....)$$
$$\Delta F_j(X) = F(.....\overline{x}_j..) - F(.....x_j..)$$
$$\Delta F_{ij}(X) = F(..\overline{x_i}..\overline{x}_j..) - F(..x_i..x_j..)$$

avec $\overline{x_i} = 1 - x_i$ et $..\overline{x_i}.....$ est le vecteur X dans lequel on a changé uniquement la variable x_i en $\overline{x_i}$. Si $\Delta F_{ij}(X) \neq \Delta F_i(X) + \Delta F_j(X)$, alors il existe une interaction non-linéaire entre les variables x_i et x_j et on considère qu'elles appartiennent à la même sous-fonction. Cette information est ensuite utilisée pour construire des opérateurs génétiques respectant les briques élémentaires et les combinant. L'étude comparative montre expérimentalement une meilleure efficacité de LINC par rapport à hBOA sur les problèmes onemax, $trap_3$ et $trap_5$. Une analyse de la complexité respective des deux approches est ensuite réalisée qui explique ce phénomène. Le paramètre principal de la complexité de LINC est en $O(Tp \cdot n^2)$ nombre d'évaluations de la fonction alors qu'il est en $O(k \cdot n^3)$ pour hBOA mais pour seulement $O(Tp \cdot \sqrt{n})$ évaluation de la fonction. Dans les problèmes pour lesquels la fonction peut s'évaluer rapidement, LINC a un avantage sur hBOA. Le papier montre finalement expérimentalement, qu'en augmentant artificiellement le temps de calcul de la fonction objectif, on arrive à une situation où hBOA découvre le maximum plus rapidement que LINC.

2.2.2.3 Discussion

Les diverses techniques d'optimisations combinatoires réagissent différemment suivant la structure et la complexité du problème considéré. D'une manière générale il y a deux grands types de problèmes d'optimisation combinatoire : ceux pour lesquels on connaît explicitement tout ou partie de la fonction objectif et ceux pour lesquels on ne connaît rien. La majeure partie des techniques de recherche opérationnelle s'adresse aux problèmes de première catégorie et est particulièrement efficace pour exploiter ces informations pour résoudre le problème. Les algorithmes d'exploration par échantillonnage sont conçus pour traiter les problèmes de la deuxième catégorie. On peut remarquer que dans ce dernier cas les notions de fonctions trompeuses et d'épistasie sont au cœur de la discussion sur l'efficacité des méthodes par échantillonnage. Une nouvelle branche de cette discipline est apparue au cours des années 90 cherchant à découvrir automatiquement les dépendances entre les variables des problèmes à résoudre pour

baisser la complexité de ces problèmes. Les algorithmes de cette branche ont maintenant fait leur preuve sur toute une classe de problèmes difficiles mais ils sont toujours sujets à un grand nombre de contraintes. Le nombre maximum de dépendances exploitables reste limité du fait du facteur exponentiel lié à ce nombre dans la complexité de ces méthodes. Les modèles construits pour découvrir les dépendances ne sont pas exacts du fait de l'utilisation d'heuristiques lors de leur construction. Les problèmes doivent être strictement ou quasi-strictement décomposables (pas ou peu de chevauchement entre les blocs de dépendances entre variables).

Mon travail et celui de plusieurs de mes étudiants ont porté ces dernières années en partie sur la compréhension et l'analyse de ces concepts et sur le développement de nouvelles techniques pour améliorer l'efficacité de ce type d'algorithme. Ce sont ces travaux que je présente dans la section suivante.

2.3 Résultats personnels

2.3.1 Un modèle parallèle hiérarchique de programmation génétique

2.3.1.1 Introduction

La *programmation génétique* (Banzhaf W. et al., 1998; Garey M.R. and Johnson D.S., 1979; Koza J.R., 1992) est une technique pour découvrir automatiquement des programmes permettant de résoudre des problèmes difficiles. Ces programmes sont représentés par des arbres eux-mêmes composés de fonctions et de terminaux[12] appropriés au problème donné. La programmation génétique est un algorithme évolutionniste qui va explorer l'espace des programmes possibles pour découvrir celui ou ceux qui sont les plus performants pour résoudre le problème. La fonction de fitness est alors une mesure de la capacité d'un programme à résoudre le problème et un chromosome est un des programmes de la population. Le calcul de la fonction de fitness nécessite en général d'exécuter sur le problème le programme à évaluer. Le processus d'optimisation commence par la génération d'une population aléatoire contenant *Tp* programmes. Cette population va ensuite évoluer au travers de générations consistant en une sélection des meilleurs programmes et en l'application d'opérateurs génétiques pour les modifier. La sélection et les opérateurs génétiques sont les mêmes que pour les algorithmes génétiques classiques : sélection proportionnelle au fitness, sélection par tournois, crossover, mutation…

La programmation génétique comporte plusieurs inconvénients par rapport aux algorithmes génétiques. Le premier vient du temps nécessaire à l'évaluation d'un programme qui est souvent beaucoup plus important que le temps de calcul de la fonction de fitness des algorithmes génétiques et cela d'autant plus que la programmation génétique est souvent utilisée pour faire de l'apprentissage automatique ce qui implique d'évaluer chaque programme sur tout un ensemble de données d'apprentissage. De plus, la structure des chromosomes (des arbres) ainsi que les opérateurs permettant de les

[12] Ces terminaux sont souvent des constantes ou les variables du problème à résoudre.

modifier font que la taille des chromosomes a tendance à augmenter considérablement au cours de la recherche. Cela augmente d'autant plus le temps nécessaire à leur évaluation et d'un point de vue pratique pose des problèmes de taille mémoire pour stocker la population de chromosomes. La parallélisation de ces algorithmes pourrait donc être une approche pertinente pour limiter ce type d'effet. Cela paraît d'autant plus évident que ce genre d'algorithmes, tout comme les algorithmes génétiques, se prête particulièrement bien à la parallélisation du fait de sa décomposition immédiate en calculs indépendants de la fitness de chaque programme. La parallélisation des algorithmes génétiques est un domaine bien étudié (Lin S.C. et al., 1994; Stracuzzi D.J., 1998; Tongchim S. and Chongstitvatana P., 2002) et ces travaux sont directement applicables à la programmation génétique (Fernandez F. et al., 2000; Golubski W., 2002). Elle peut être classée selon trois modèles différents :

1. La parallélisation *maître-esclave* (master-slave). Ce modèle utilise une population unique globale et c'est l'évaluation qui est faite en parallèle sur plusieurs processeurs. Le processus maître réalise le corps de l'algorithme évolutionniste. Durant l'étape d'évaluation chaque processus esclave collecte un même nombre de chromosomes à partir du maître, puis les évalue et retourne leur valeur de fitness au maître. L'étape de reproduction peut être également effectuée en parallèle. Ce modèle implique un nombre important d'échanges entre les différents processus et est donc plus particulièrement dédié aux ordinateurs à mémoire partagée.

2. La parallélisation à *grain fin* (fine-grained). Dans ce modèle (Manderick B. and Spiessens P., 1989) , la population est divisée en un grand nombre de petites sous-populations. Chacune d'elles est associée à un processus indépendant. Les sous-populations sont spatialement distribuées sur une grille de processeurs et elles n'interagissent que localement avec leur voisinage direct. Ce modèle est particulièrement adapté aux machines constituées d'un grand nombre de processeurs simples connectés par des liaisons à grandes vitesses (machines massivement parallèles).

3. La parallélisation à *grain épais* (coarse-grained) (Cantu-Paz E. and Goldberg D.E., 1999). Ici, la population est divisée en un petit nombre de sous-populations. Chacune d'elles applique un processus évolutif indépendant complet avec ses propres paramètres (pression de sélection, probabilité des opérateurs, taille de la sous-population…). A intervalles réguliers ou lors de la réalisation d'un critère particulier, elles envoient un nombre fixé de chromosomes à une ou plusieurs autre(s) sous-population(s). En retour, un nombre identique de chromosomes sont reçus en provenance d'une ou plusieurs autre(s) sous-population(s). Cette migration peut intervenir soit de façon asynchrone, soit après synchronisation de toutes les sous-populations. Ce modèle est aussi connu sous le nom de *modèle en îlots* (island model). Une de ces propriétés principales, outre la parallélisation elle-même, est de limiter la convergence prématurée[13] du processus global d'exploration en préservant par isolement la diversité des solutions représentées

[13] C'est-à-dire la stagnation de l'exploration autour d'un maximum local.

dans la population totale. Ce concept est inclus dans celui de *niches écologiques* (Goldberg D.E., 1989a; Holland J.H., 1971) qui comprend aussi les techniques de *partage de fitness* (fitness sharing) (Goldberg D.E. and Richardson J., 1987) et de *rassemblement* (crowding) (Cavicchio D.J., 1970; Mengshoel O.J. and Goldberg D.E., 1999).

Un autre problème important en programmation génétique est celui du *gonflement* (bloat) (De Jong E.D. et al., 2001; Soule T. and Foster J.A., 1999). Ce problème apparaît lorsque la taille des arbres solutions augmente au cours du processus d'exploration et qu'elle devient plus grande que ce qui serait nécessaire pour coder cette solution de manière compacte. Cela arrive quand une partie de l'arbre n'intervient plus dans le calcul de la valeur de fitness de la solution ou qu'elle joue un rôle neutre dans ce même calcul[14]. La conséquence de ce phénomène est l'augmentation, parfois considérable, du temps de calcul nécessaire à l'évaluation de l'arbre et de l'espace mémoire utilisé pour le stocker. Cela réduit de plus fortement la probabilité que les opérateurs génétiques soient appliqués à une partie informative de l'arbre.

Nous présentons dans cette section un nouveau modèle de programmation génétique parallèle à grain épais appelé *modèle pyramidal* (Frey J. et al., 2004; Soule T. and Foster J.A., 1999). Il est basé sur une parallélisation hiérarchisée en îlots et s'efforce de limiter les différents problèmes cités précédemment.

2.3.1.2 La méthode

Le modèle pyramidal (PyM) utilise une topologie de communication entre sous-populations plus efficace que celle de la programmation génétique parallèle à grain épais. Dans PyM, les sous-populations sont distribuées dans plusieurs niveaux superposés. Le nombre de sous-populations contenues dans chaque niveau augmente avec la profondeur dans la pyramide. Les communications se font entre deux niveaux consécutifs et de façon unidirectionnelle. Ainsi, chaque sous-population envoie des programmes à une sous-population du niveau directement supérieur et reçoit des programmes de sous-populations situées dans le niveau directement inférieur. Ce modèle topologique peut être vu comme une communication pyramidale entre sous-populations (La figure 2 montre cette topologie de communication). Les paramètres principaux de chaque sous-population sont définis en fonction de la profondeur du niveau auquel elles appartiennent. Plus cette profondeur est élevée et plus la taille de la sous-population augmente, plus le nombre d'itérations de l'algorithme de programmation génétique diminue, plus la probabilité d'application des opérateurs génétiques est élevée et plus le temps d'évaluation de la fitness des chromosomes diminue. Dans le cadre de l'application de notre méthode à l'apprentissage automatique, l'ensemble d'apprentissage n'est disponible dans son intégralité que pour la sous-population de niveau le plus haut. Les sous-populations des niveaux inférieurs n'ont accès qu'à un échantillon de l'ensemble d'apprentissage, échantillon d'autant plus petit que le niveau de la sous-population est bas. De cette façon,

[14] Des fonctions successives qui s'annulent entres elles ou qui empêchent l'exécution d'une partie du code de l'arbre par exemple.

les sous-populations les plus basses explorent rapidement l'espace de recherche sans risque de rester bloquées sur un maximum local. Elles envoient de plus des solutions prometteuses aux sous-populations des niveaux supérieurs qui exploitent ces informations. De ce fait, cette topologie de communication permet de séparer et d'équilibrer les phases d'explorations et d'exploitations de l'algorithme de programmation génétique, problème bien connu dans le contexte des métaheuristiques (Michalewicz Z. and Fogel D., 2000). L'exploration se fait donc dans les niveaux les plus bas, dans lesquels l'évaluation de la fitness des chromosomes se fait sur un petit échantillon de l'ensemble d'apprentissage et nécessite donc peu de temps de calcul. Ainsi ce processus d'exploration peut découvrir rapidement des parties de l'espace de recherche prometteuses. Les meilleures de celles-ci sont envoyées dans les niveaux supérieurs où elles sont combinées et où leur fitness est évalué sur un échantillon plus significatif car plus divers.

Nous utilisons une approche d'*optimisation multi-objectifs* (Coello Coello C.A. et al., 2002), pour résoudre le problème du gonflement. L'optimisation multi-objectifs est la recherche de solutions optimales selon plusieurs critères simultanément. A chacun de ces critères correspond une fonction objectif à optimiser. On notera $F^* = \{F_1, F_2, \ldots, F_l\}$ l'ensemble des l fonctions objectif à optimiser simultanément. Dans le cas où l'on ne peut définir une mesure unique regroupant ces différents critères, on peut essayer de découvrir la solution optimale *au sens de Pareto*. Une solution optimale X_p au sens de Pareto pour un ensemble de fonction F^* signifie qu'il n'existe aucun autre point X de l'espace de recherche pour lequel :

$$\exists\, F_i \in F^* \text{ tel que } F_i(X) > F_i(X_p) \text{ et } \neg\exists\, F_j \in F^* \text{ tel que } F_j(X) < F_j(X_p).$$

Dans notre approche, nous considérons deux objectifs simultanés, l'objectif principal (obj_1) qui correspond au problème que l'on cherche à résoudre et l'objectif de taille (obj_2) qui correspond au nombre de nœuds de l'arbre dont on veut estimer la fitness. Nous utilisons de plus la dominance de Pareto. Dans notre cas, un point X_1 domine un point X_2 si :

$$obj_1(X_1) > obj_1(X_2) \text{ et } obj_2(X_1) < obj_2(X_2)$$

Un point X est considéré comme non dominé s'il n'existe aucun autre point qui le domine. Au cours du processus de l'algorithme de programmation génétique nous maintenons pour chaque sous-population une liste des solutions rencontrées qui ne sont dominées par aucune des autres solutions rencontrées. Quand un nombre donné de générations a été effectué pour une sous-population donnée (nombre dépendant du niveau dans lequel se trouve la sous-population), tous les programmes contenus dans la liste sont envoyés à la sous-population du niveau supérieur et un nouveau processus de programmation génétique commence pour la première sous-population. La réception de programmes est vérifiée à chaque étape d'évaluation dans chacune des sous-populations et les programmes reçus sont placés dans une file d'attente. A chaque étape de reproduction, certains programmes sont retirés de la file et intégrés à la population sans les modifier. De ce fait la communication au sein de notre pyramide le processus se fait

de manière totalement asynchrone. Le processus global PyM s'arrête lorsque la population du niveau le plus haut a réalisé un nombre de générations fixé.

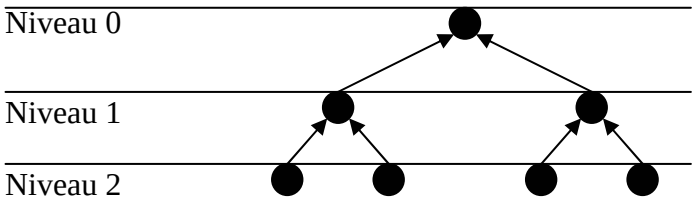


Figure 2. La topologie de communication du modèle pyramidal de programmation génétique parallèle.

2.3.1.3 Application

Ces dernières années, le séquençage massif de génomes complets a permis la détermination de la séquence de plusieurs dizaines de milliers de nouvelles protéines dont beaucoup d'entre elles n'ont pas de fonction connue. La découverte expérimentale de la fonction de nouvelles protéines étant un travail particulièrement long (plusieurs années) et coûteux, la prédiction, ou tout au moins l'aide à la prédiction, automatique de leurs fonctions est devenu indispensable. Il est cependant difficile de déterminer la fonction d'une protéine en ayant pour seule information sa séquence primaire. La recherche d'homologie entre séquences par similarité avec des séquences de fonctions connues est la méthode la plus couramment utilisée pour prédire la fonction de nouvelles protéines (Mullan, 2002). La similarité est mesurée par l'alignement des séquences en maximisant un score de similarité. De multiples outils existent pour réaliser des alignements aussi bien entre deux séquences (Altschul S.F. et al., 1990; Lipman D.J. and Pearson W.R., 1985) qu'entre plusieurs séquences complètes (Morgenstern B., 1999; Thompson J.D. et al., 1994) ou des portions de séquences simultanément (Hernandez D. et al., 2004; Lawrence C.E. et al., 1993). Ces approches par homologie ne sont fiables que si la similarité est forte. Une autre technique est utilisée basée uniquement sur l'analyse de la séquence primaire (Jensen L.J. et al., 2002). Cette méthode repose sur le fait qu'un grand nombre de propriétés de la séquence primaire (appelées aussi attributs de séquence) dépendent de la fonction de la protéine. Une approche de prédiction de fonction basée sur la recherche de tels attributs[15] dans les séquences primaires peut dès lors être envisagée.

Notre méthode intègre 21 attributs calculés directement à partir de la séquence primaire. Elle a pour but d'apprendre une fonction de ces attributs permettant de

[15] Ces attributs comprennent, entre autres, la longueur, le point isoélectrique, la composition en acides aminés de la séquence ou des motifs fonctionnels connus.

déterminer automatiquement si une protéine participe au processus biologique de la transcription. Nous l'avons appliquée à la détection des facteurs de transcription mais le même schéma pourrait tout aussi bien être appliqué pour la prédiction d'autres fonctions biologiques, pour peu que l'on dispose de données d'apprentissages fiables en quantité suffisante. La capacité de discrimination de chacun des attributs pris séparément n'est pas suffisante pour prédire si une protéine est un facteur de transcription. Par contre, une combinaison pondérée de ces facteurs peut parfaitement réaliser cette prédiction. Nous avons donc appliqué notre approche de programmation génétique parallèle pour découvrir une telle combinaison.

2.3.1.3.1 Ensemble de terminaux, ensemble de fonctions et architecture

Dans notre problème, un programme est un prédicteur de facteur de transcription. Il se compose d'opérations arithmétiques et d'opérateurs conditionnels. L'ensemble des terminaux, appelé *T*, consiste en 21 attributs et des constantes, appelées *C*, générées aléatoirement entre -10^6 et $+10^6$. L'ensemble des fonctions, appelé *F*, est composé des opérateurs arithmétiques standards (addition, soustraction, division et multiplication) et d'un opérateur conditionnel. La figure 3 donne un exemple d'un programme valide généré à partir des ensembles *T* et *F*. Pour une séquence protéique donnée, les valeurs des attributs sont calculées et utilisées pour instancier les variables terminales correspondantes du programme à évaluer. Le programme retourne un score qui est comparé à une valeur seuil dépendante du programme. Cette comparaison détermine si la protéine est classée parmi les facteurs de transcriptions ou non.

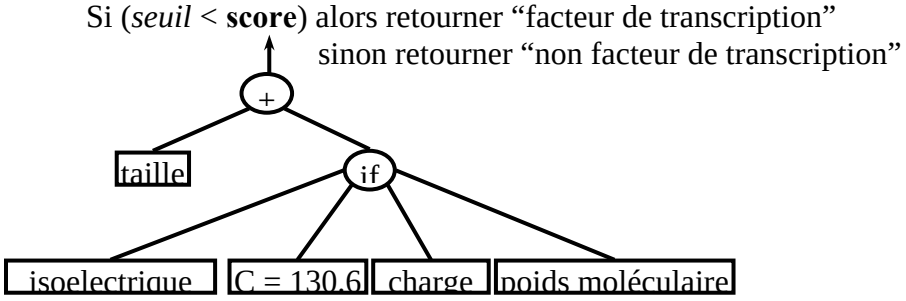


Figure 3. Un exemple de programme valide généré à partir des ensembles *T* et *F*. Le programme retourne "**score**" qui est comparé au *seuil* pour décider si la protéine correspondante est ou non un facteur de transcription.

2.3.1.3.2 Mesure de fitness et ensemble d'apprentissage

La fonction que nous cherchons à optimiser, donc la valeur de fitness des chromosomes/programmes de notre population, est la fonction qui classifie les protéines entre "facteur de transcription" et "non facteur de transcription". On a préalablement divisé notre ensemble d'apprentissage en deux : l'ensemble positif contenant les

protéines étant des facteurs de transcription et l'ensemble négatif contenant les protéines n'étant pas des facteurs de transcription. On calcule la fitness en comparant les valeurs retournées par le programme à évaluer pour les protéines de l'ensemble positif avec les valeurs retournées pour les protéines de l'ensemble négatif. Nous utilisons le test de Kolmogorov-Smirnov (test-KS) qui permet de déterminer si deux distributions sont significativement différentes. Ce test a l'avantage de ne faire aucune hypothèse sur la forme des distributions à comparer et ne nécessite aucun ajustement de paramètre. Le test-KS considère la déviation verticale *D* maximale entre deux distributions cumulées. Dans notre cas, nous cherchons à maximiser la différence entre la distribution des scores des protéines positives et négatives. Nous cherchons donc à obtenir la plus grande valeur de *D* possible. En effet, plus celle-ci est grande et plus les deux distributions sont séparées, donc plus le programme calcule un score discriminant et donc plus la prédiction réalisée sera fiable. La valeur *D* est comprise entre 0 et 1. La fitness d'un programme est ainsi calculée selon les étapes successives suivantes. Premièrement, le score donné par le programme est calculé pour toutes les protéines positives. Puis la liste des scores obtenus est triée par ordre croissant. Les deux mêmes étapes sont appliquées aux protéines négatives. Le test-KS est ensuite appliqué aux deux listes précédentes. La valeur *D* obtenue est alors la valeur de fitness du programme. Le programme peut ainsi être utilisé comme prédicteur en prenant *D* comme valeur *seuil*. Un exemple du calcul de *D* est donné dans la figure 4.

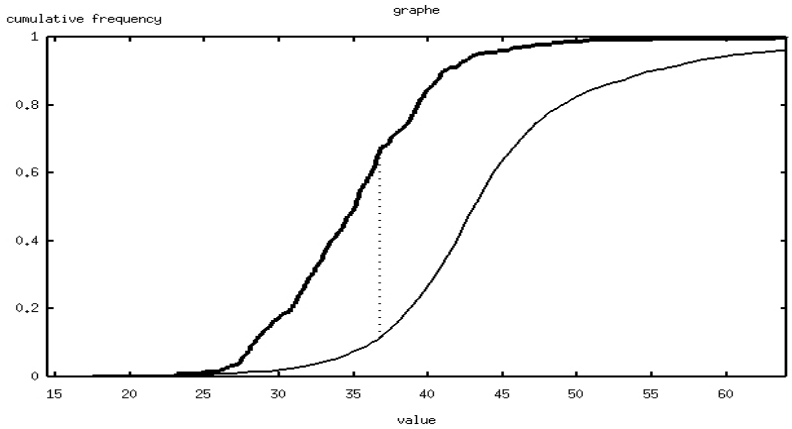


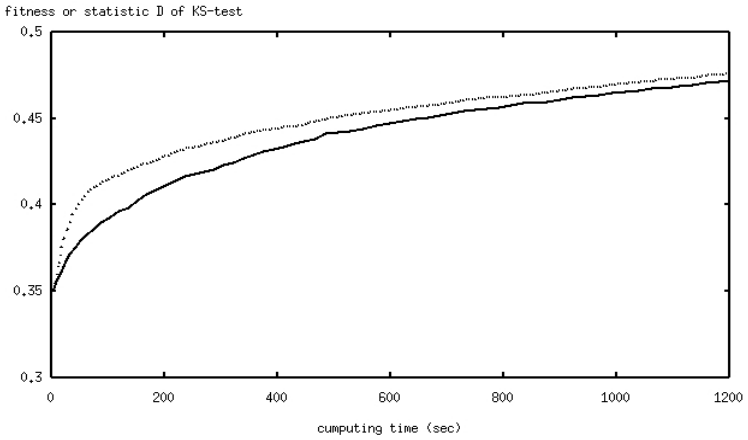
Figure 4. Distribution cumulée du score de 460 protéines positives (courbe en gras) et distribution cumulée du score de 4012 protéines négatives (courbe normale) obtenues pour un programme. La ligne pointillée représente la valeur *D* (valeur du *seuil* = 36.8).

Pour tester notre méthode, nous avons utilisé un ensemble d'apprentissage composé de protéines extraites de la base de données SWISS-PROT (Boeckmann B. et al., 2003). Cet ensemble était composé de 8024 protéines humaines négatives et 920 protéines humaines positives. Nous avons divisé les données en deux ensembles : un ensemble d'apprentissage utilisé par l'algorithme de programmation génétique pour apprendre une fonction de score et un ensemble de contrôle utilisé après l'apprentissage pour tester le programme appris. Les deux ensembles sont constitués de 4013 protéines négatives et 460 protéines positives.

2.3.1.4 Résultats et discussion

2.3.1.4.1 Test de l'optimisation multi-objectifs

Nous avons testé l'efficacité de l'approche par optimisation multi-objectifs. Pour cela nous avons comparé la qualité des solutions générées par deux algorithmes ne différant que par leur fonction de sélection. Le premier algorithme, appelé *Sélection par Tournoi* (ST), sélectionne les chromosomes de la population par sélection par tournoi de taille deux[16] (Blickle T. and Thiele L., 1995). Cet algorithme n'optimise qu'un seul objectif : la discrimination de la fonction de score. Le deuxième algorithme, appelé Multi-Objectifs (MO), utilise deux critères de sélection. Nous avons choisi la méthode de sélection proposée par Fonseca et Flemming (Fonseca C.M. and Fleming P.J., 1993) pour leur algorithme Multi-Objective Genetic Algorithm. Son principe est de maximiser la fonction de fitness (ici la discrimination de la fonction de score) tout en minimisant la taille (son nombre de nœuds) du programme correspondant. Comme nous ne cherchions qu'à mesurer l'apport de l'approche multi-objectifs, nous avons utilisé deux algorithmes en mode séquentiel. Nous avons choisi de mesurer l'évolution de la fitness en fonction du temps de calcul comme critère de comparaison entre les deux algorithmes. Cela se justifie par le fait qu'elles sont très similaires, MO nécessitant même un peu plus de temps que ST pour traiter ses deux objectifs et ses listes de dominants, le gain en temps de calcul impliquant bien alors un gain en efficacité. La figure 5 montre les graphes obtenus pour les deux algorithmes pour des populations de taille 400. Des résultats similaires, non présentés ici, ont été obtenus pour d'autres tailles de populations. Ces résultats sont des moyennes réalisées sur 100 exécutions de chacun des algorithmes.



[16] Cela consiste à sélectionner deux par deux avec remise les chromosomes de la population à la génération *t* et à choisir parmi les deux celui de plus haute fitness pour le mettre dans la population de la génération *t + 1*.

Figure 5. Fitness des solutions générées par les algorithmes MO (courbe en pointillé) et par ST (courbe pleine) en fonction du temps d'exécution pour une population de 400 chromosomes. Résultats moyens sur 100 exécutions.

L'algorithme MO génère, à temps égal, des solutions de légèrement meilleure qualité que l'algorithme ST. De plus la taille des programmes obtenus par MO est nettement plus faible que celle des programmes obtenus par ST. En effet, à la fin du processus d'apprentissage de l'algorithme génétique la taille moyenne des 400 chromosomes est de 87 nœuds pour MO et de 801 pour ST. Ainsi, à temps égal, MO génère des solutions de meilleure qualité et plus parcimonieuses que ST. La différence de temps d'exécution peut en grande partie s'expliquer par le gain de temps réalisé pour évaluer des solutions de plus petites tailles. Mais il est cependant intéressant de constater l'efficacité de l'approche multi-objectifs pour produire des solutions compactes avec au moins la même capacité de discrimination.

2.3.1.4.2 Apport de l'approche parallèle

Nous avons testé les performances de notre méthode de programmation génétique parallèle hiérarchique. Pour cela nous avons comparé les qualités des solutions obtenues par une version série et une version parallèle de notre algorithme. La version parallèle étant distribuée sur plusieurs processeurs, pour que les résultats restent comparables, nous les avons basés sur le nombre total de protéines évaluées au cour de l'apprentissage. Pour l'implémentation de la version parallèle nous avons utilisé la librairie de communication MPI (Pacheco P.S., 1997). Pour l'algorithme série, nous avons utilisé une population de 400 chromosomes et un ensemble d'apprentissage de 4472 protéines. Pour la version parallèle, nous avons utilisé quatre sous-populations connectées suivant la topologie hiérarchique décrite précédemment. La première sous-population, de niveau 0, contient 200 chromosomes et 4472 protéines. Les trois autres sous-populations sont de niveau 1, chacune d'elles comprenant 500 chromosomes et un ensemble d'apprentissage de 500 protéines. Elles envoient leurs solutions non dominées à la sous-population de niveau 0 toutes les 20 générations. La figure 6 montre les résultats obtenus en réalisant une moyenne sur 100 exécutions.

Nous pouvons remarquer que l'algorithme parallèle nécessite moins d'évaluations de protéines pour obtenir des solutions de qualité équivalente à celles de l'algorithme parallèle. Ce résultat étant indépendant de la distribution du calcul sur plusieurs processeurs, puisque mesuré sur le nombre d'évaluations effectuées, il confirme l'intérêt de l'approche des niches écologiques. Le nombre de paramètres à gérer avec l'approche parallèle (paramètres de l'algorithme évolutif dans chaque sous-population, taille des sous-populations, modalité de communication entre sous-populations…) étant encore plus important que dans la version série, il conviendrait de faire de plus amples tests dans de multiples conditions pour valider rigoureusement ces résultats. Les études réalisées sur la taille de la population nécessaire pour traiter les problèmes avec épistasie (Pelikan M.,

2002) laisseraient toutefois supposer que diviser la population ne serait pas le meilleur moyen de profiter des propriétés des niches écologiques. Cela réduit en effet la taille de l'échantillon de l'espace de recherche disponible dans chaque sous-population et risque d'empêcher toutes ces sous-populations de découvrir les dépendances significatives. Etant donné le temps de calcul important nécessaire pour l'approche par partage de fitness (évaluation de la distance entre tous les couples de chromosomes de la population) et sa difficulté d'application à la programmation génétique, la méthode par regroupement pourrait être une stratégie préférable. La parallèlisation serait toutefois toujours utile dans un modèle maitre-esclave puisque la fonction de fitness est longue à évaluer.

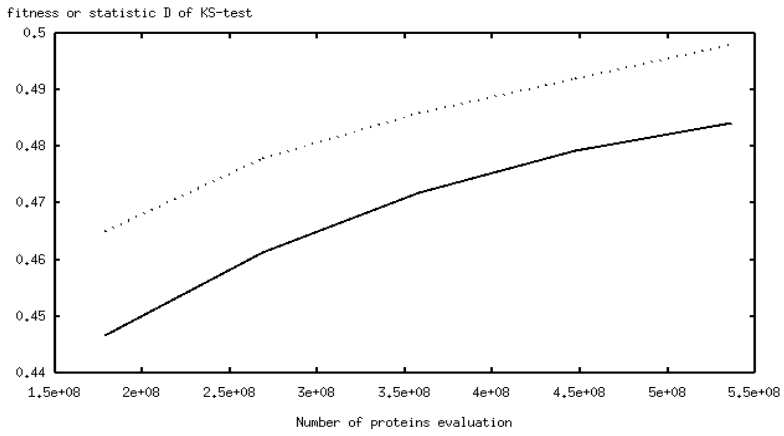


Figure 6. Fitness des meilleures solutions générées par l'algorithme série (courbe pleine) et par l'algorithme parallèle (courbe pointillée) en fonction du nombre de protéines évaluées. Les résultats sont une moyenne sur 100 exécutions.

2.3.2 Auto-adaptation des opérateurs

2.3.2.1 Introduction

Nous nous sommes intéressés à l'optimisation du comportement d'exploration de l'algorithme génétique classique. Un des problèmes inhérents à cette approche est le grand nombre de paramètres à spécifier pour déterminer le mode d'exploration de l'algorithme. Il y a différents types de sélections possibles avec différents niveaux de pression de sélection, différents opérateurs de mutation et de crossover, pour chacun d'eux une probabilité d'occurrence et la taille de la population. La valeur optimale de ces différents paramètres dépend de plus du problème à traiter. De nombreuses publications (De Jong K.A., 1975; Sharpe O.J., 2000) se sont intéressées à la détermination des valeurs optimales de ces paramètres. Malheureusement, la structure de l'espace de recherche va rendre plus ou moins efficace chacun de ces opérateurs, le degré d'épistasie ou le fait que la fonction soit trompeuse va influer sur la taille de la population… Il paraît peu réaliste d'étudier toutes les situations possibles pour déterminer dans chaque cas quel est le meilleur choix à faire pour ces paramètres.

Dans les algorithmes génétiques classiques les paramètres restent constants ou sont modifiés de façon prédéfinie en fonction du nombre de générations effectuées au cours du déroulement du programme. Les méthodes cherchant à découvrir par essais-erreurs la bonne valeur des paramètres pour une application donnée sont très consommatrices de ressources et peu généralisables. Davis a initié la discussion sur le choix des paramètres et sur une méthode permettant de faire ce choix automatiquement pendant la simulation (Davis L., 1991). Son principe repose sur le changement des probabilités d'occurrences des opérateurs en fonction de leur contribution à la découverte des meilleures solutions. Ce type d'approche a été appelé *approche adaptative*. Elle a été appliquée avec succès par Notredame et Higgins (Notredame C. and Higgins D.G., 1996) pour des algorithmes génétiques dédiés à l'alignement multiple de séquences biologiques. Plusieurs méthodes adaptatives pour les algorithmes génétiques sont présentées dans (Herrera F. and Lozano M., 1996).

Nous présentons dans cette section un nouvel algorithme, appelé *Highway*, conçu selon le principe adaptatif et basé sur une nouvelle mesure, la *redondance*, permettant de déterminer quand et comment adapter les paramètres de l'algorithme (Martin O. et al., 2003). Nous utilisons également une représentation de l'espace exploré nous permettant de calculer efficacement la mesure de redondance et d'éviter de multiples évaluations de fitness d'un même point. Le but de cette approche est de "rentabiliser" au maximum le temps de calcul en évitant de perdre du temps avec des opérations peu utiles à un moment donné de l'exploration. Elle cherche également à éviter le plus possible la dépendance aux choix initiaux des paramètres. Une version préliminaire de Highway avait été testée sur le problème de la découverte de motifs dans les séquences biologiques en association avec l'outil MoDEL présenté dans la section 3.2.2 montrant une légère amélioration des performances (Hernandez D. et al., 2002).

2.3.2.2 La redondance

Dans les algorithmes génétiques classiques, les opérateurs génétiques sont souvent appliqués stochastiquement les uns après les autres aux chromosomes de la population. La première différence, dans l'algorithme Highway, est que, pour tous les couples choisis par la sélection, un opérateur au plus sera appliqué. Le choix de cet opérateur est fait par sélection par roue de loterie, c'est-à-dire un choix probabiliste uniforme sur la somme normalisée à 1 des probabilités des opérateurs utilisés. Ces probabilités sont modifiées régulièrement en fonction de la diversité de la population et de leur valeur respective de redondance. La diversité est une mesure de la quantité de variabilité au sein de la population de chromosomes qui peut être calculée de différentes façons. Nous cherchons, par son intermédiaire, à contrôler le niveau de diversité de la population pour éviter une convergence prématurée (sur un maximum local) de l'algorithme.

Le critère sur lequel nous basons notre approche est la mesure de redondance, qui est une mesure complémentaire de la notion de diversité. Etant donné *no* opérateurs

génétiques {op_1, op_2, …, op_{no}}, la mesure de *redondance instantanée* r_i associée à chaque opérateur op_i et calculée à chaque génération est définie par :

$$r_i = 1 - \left(\frac{new_i}{tot_i} \right)$$

avec tot_i le nombre total de chromosomes sur lesquels cet opérateur a été appliqué à la génération courante, new_i le nombre de chromosomes, parmi ceux générés par l'application de l'opérateur, qui n'ont jamais été rencontrés dans les générations précédentes. Pour avoir en permanence accès à cette information, nous conservons une représentation compacte de l'ensemble des chromosomes rencontrés. Cette représentation se présente sous la forme d'un arbre des chromosomes. A chaque fois qu'un opérateur modifie une solution, la solution est recherchée dans l'arbre (en temps proportionnel à n, longueur du chromosome). Si la solution est déjà présente dans l'arbre, la redondance de l'opérateur qui l'a générée augmente puisque tot_i augmente et pas new_i. Dans le cas contraire, tot_i et new_i augmentent (et donc la redondance diminue) et la solution est ajoutée à l'arbre. L'utilisation de la structure d'arbre permet de savoir à n'importe quelle étape de la simulation quelles solutions ont déjà été rencontrées et quelle est la capacité de chaque opérateur à générer de la diversité à cette étape. Cette structure peut donc être vue comme une représentation de l'espace exploré. La figure 7 montre un exemple d'un arbre contenant 11 chromosomes binaires de taille 6. La complexité de cette structure augmente au plus linéairement avec le nombre de chromosomes différents rencontrés.

La mesure de redondance instantanée peut aussi être calculée pour la population totale plutôt que pour un opérateur. Dans ce cas elle est appelée *r* et *new* est le nombre de nouveaux chromosomes pour la population complète à cette génération et *tot* est le nombre total de chromosomes générés à cette génération.

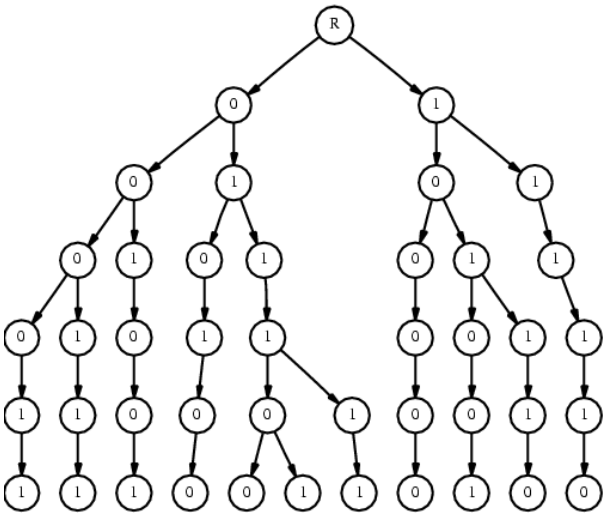


Figure 7. Une structure en arbre contenant 11 chromosomes binaires de longueur 6. Le nœud racine *R* est un nœud virtuel utilisé pour initier les parcours de l'arbre.

Bien que la mesure de redondance r_i soit utile, elle peut être considérée comme instable car il peut y avoir des variations importantes d'une génération à l'autre. De manière à lisser cette instabilité, nous avons construit une autre mesure $\bar{r}_i$ de *redondance lissée* de l'opérateur op_i basée sur une moyenne de la redondance sur une "fenêtre" de générations. Elle se définit par :

$$\bar{r}_i = 1 - \frac{1}{z}\left(\frac{\sum_{j=1}^{z} new_{ij}}{\sum_{j=1}^{z} tot_{ij}} \right)$$

avec z le nombre de générations dans la fenêtre considérée, new_{ij} le nombre de nouveaux chromosomes créés par l'opérateur op_i à la génération j et tot_{ij} le nombre total de chromosomes générés par l'opérateur op_i à la génération j. On définit de la même façon une mesure $\bar{r}$ de redondance lissée de la population pour tous les chromosomes de la population au cours des z générations.

2.3.2.3 Adaptation des opérateurs

Un critère simple pour déterminer si les probabilités d'occurrences des opérateurs doivent être adaptées est de détecter que l'exploration stagne sur un maximum local. Par exemple, quand la fitness maximum découverte jusqu'à présent n'a pas changé depuis un nombre donné de générations on peut décider d'adapter les probabilités (méthode 1). Nous avons également défini un critère plus complexe (méthode 2) basé sur les observations suivantes :

(1) la pente de la régression linéaire effectuée sur les valeurs successives (c'est-à-dire sur plusieurs générations) de fitness maximum. Si elle est plus faible qu'une valeur seuil choisie, la fitness maximum est considérée comme étant constante (condition 1).

(2) le coefficient de corrélation de la droite de régression définie en (1). Si ce coefficient est plus petit qu'une valeur seuil choisie, la fitness maximum est considérée comme instable (condition 2).

(3) un coefficient de diversité d définie par :

$$d = 1 - \left(\frac{fit_{moyen}}{fit_{\max}} \right)$$

avec fit_{moyen} le fitness moyen de la population et fit_{max} le fitness maximum courant de la population. Lorsque d est inférieur à une valeur seuil choisie le coefficient de diversité est dit insuffisant (condition 3).

En combinant ces trois conditions il est possible de déterminer si la simulation stagne sur un maximum local. Par exemple, si les conditions 1 et 2 sont vraies, cela signifie que la fitness maximum est à la fois constante (en moyenne sur plusieurs générations) et instable. On peut considérer dans ce cas que la simulation génère trop de diversité (peut-être du fait d'un opérateur de type mutation avec une probabilité trop grande) et détruit rapidement toutes les solutions prometteuses. Par contre, si les conditions 1 et 3 sont vraies, cela signifie que la simulation a besoin de générer plus de diversité.

Alors que pour la méthode 1 la seule action possible est d'augmenter la diversité quand la fitness maximum ne change pas, avec la méthode 2 plusieurs actions sont envisageables :

(1) ne rien faire, c'est-à-dire la simulation ne génère ni trop peu ni pas assez de diversité la simulation peut donc continuer telle quelle.

(2) augmenter la diversité.

(3) diminuer la diversité.

L'utilisation de conditions multiples et les trois actions possibles de la méthode 2 donnent la possibilité de construire des stratégies complexes de réaction aux différents états de la simulation de façon à corriger ou amplifier les phénomènes détectés suivant les besoins du moment. Par exemple la stagnation de la valeur du fitness maximum en début ou en fin de simulation ne provoquera pas les mêmes changements sur les probabilités des opérateurs.

Une fois l'action à entreprendre décidée, il faut déterminer comment effectuer les changements de probabilités. On appelle $p_i(t)$ la probabilité d'occurrence de l'opérateur op_i à la génération t pour tous les chromosomes. Comme nous voulons qu'un seul opérateur au maximum soit appliqué à chaque chromosome, nous devons maintenir la propriété (propriété 1) suivante :

$$\sum_{i=1}^{no} p_i(t) = 1$$

La probabilité de l'opérateur op_i à la générateur $t + 1$ est définie par :

$$p_i(t+1) = p_i(t)\Delta p_i$$

avec Δp_i la variation de probabilité de la génération *t* à la génération *t* + 1 pour l'opérateur op_i. Δp_i est définie par :

$$\Delta p_i = \frac{\bar{r}}{r_i} \quad \text{quand on cherche à augmenter la diversité.}$$

$$\Delta p_i = \overline{rr_i} \quad \text{quand on cherche à diminuer la diversité.}$$

Quand on cherche à augmenter la diversité, les opérateurs avec une valeur de redondance inférieure à la moyenne sont renforcés et les opérateurs avec une valeur de redondance supérieure à la moyenne vont être diminués. Quand on cherche à diminuer la diversité on a la situation inverse, on renforce les opérateurs de redondance élevée et on diminue les opérateurs de redondance faible. Pour maintenir la propriété 1, une fois toutes les probabilités transformées, elles doivent être ré-étalonnées de façon à ce que leur somme reste égale à un. Pour éviter de perdre définitivement un opérateur en amenant sa probabilité à 0, nous utilisons une valeur inférieure au-delà de laquelle les probabilités ne peuvent plus baisser.

Le nombre de paramètres supplémentaires que nous utilisons peut paraître important (plusieurs valeurs seuils, taille de fenêtre…). Nous espérons cependant que ces paramètres soient beaucoup plus stables d'une application à l'autre car ils ne portent que sur la manière dont l'algorithme doit se comporter pour s'adapter à l'état de la population et de l'exploration. Ils doivent de plus permettre un choix beaucoup moins strict des paramètres classiques des algorithmes génétiques puisque ces paramètres vont s'adapter à l'application.

2.3.2.4 Fonctions de tests et résultats

Les fonctions de tests que nous avons utilisées, *3-deceptive* et *6-bipolar* sont des fonctions trompeuses classiques définies dans (Goldberg D.E. et al., 1992). Nous avons comparé le comportement de notre algorithme, Highway, sur ces deux fonctions avec celui d'un algorithme génétique classique (mutation ponctuelle, crossover à un point) et avec BOA (Pelikan M. et al., 1999), conçu spécifiquement pour résoudre ce type de problèmes. Highway comporte trois opérateurs génétiques : la mutation ponctuelle, le crossover à un point et le crossover à deux points. Les tailles de la population *Tp* des trois algorithmes croissent linéairement avec la taille du problème *n* (20 * *n* pour la fonction 3-deceptive et 15 * *n* pour la fonction 6-bipolar). Nous avons défini 11 classes de tailles de problèmes entre 30 et 90 variables binaires, un tous les 6 variables. Pour chaque classe de problème nous avons mesuré les résultats en réalisant une moyenne sur 100 exécutions indépendantes de chaque programme. Les conditions d'arrêt des algorithmes sont la découverte de la solution optimale ou le nombre de générations effectuées est égal à 500. Lorsque l'un des algorithmes ne trouve pas la meilleure solution dans les 500 générations qui lui sont imparties, les résultats ne sont pas pris en compte dans les moyennes. On

mémorise toutefois ces échecs qui sont présentés dans la table 1. Tous les paramètres communs à l'algorithme génétique classique et à Highway ont été choisis avec des valeurs identiques pour que la comparaison fasse ressortir le plus clairement possible les différences entre les approches. Pour BOA, nous avons choisi de prendre ses paramètres par défaut qui se sont montrés les plus efficaces sur ces tests.

La plupart des comparaisons réalisées sur les différentes variantes des algorithmes génétiques sont basées sur le nombre de points de l'espace de recherche évalués durant la simulation. Bien qu'important, ce critère n'est pas le seul à prendre en considération. Le temps de calcul total, dans des conditions identiques, est aussi un facteur important, en particulier lors de comparaisons avec des approches PMBGA dans lesquelles la construction du modèle statistique peut nécessiter une part non négligeable du temps de calcul (Munetomo M. et al., 2004). C'est la raison pour laquelle nous donnons nos résultats selon les deux mesures. La machine utilisée dans toutes les expériences était un Pentium III à 100GHz.

2.3.2.4.1 La fonction 3-deceptive

Les figures 8 et 9 montrent comment se comportent les trois méthodes en terme de nombre de points de l'espace de recherche évalués et en terme de temps de calcul. Pour le premier critère, Highway a des performances légèrement moins bonnes que BOA mais qui restent dans le même ordre de grandeur. L'algorithme génétique classique a par contre besoin de beaucoup plus d'évaluations que les deux autres avec une dépendance non linéaire avec la taille du problème. Dans la figure 8, on peut constater que Highway est de loin le programme le plus performant du point de vue du temps de calcul. La complexité de la construction du réseau bayésien de BOA, qui est au moins cubique sur la taille du problème, a donc des conséquences importantes sur le comportement global du programme.

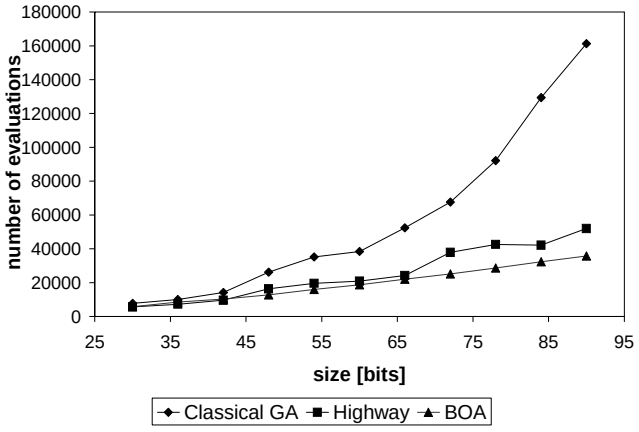


Figure 8. Nombre d'évaluations de la fonction de fitness pour la fonction 3-deceptive et pour les trois méthodes en moyenne sur 100 exécutions.

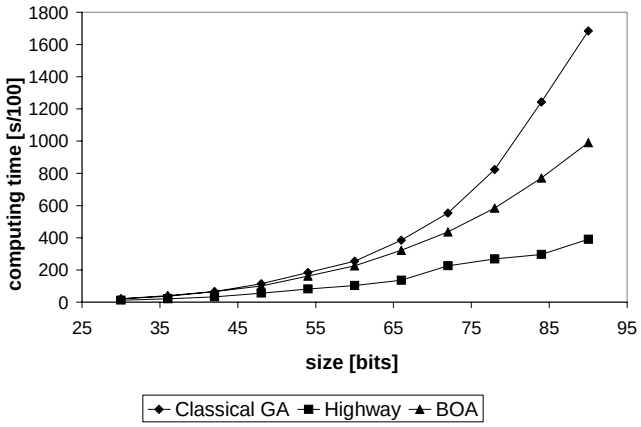


Figure 9. Temps de calcul total pour la fonction 3-deceptive et pour les trois méthodes en moyenne sur 100 exécutions.

2.3.2.4.2 La fonction 6-bipolar

La figure 10 montre le même phénomène que la figure 8 mais de manière moins prononcée, le nombre d'évaluations de la fonction de fitness de BOA augmente légèrement moins vite que celui de Highway avec la taille du problème. La figure 11 montre une fois encore que Highway est de loin plus rapide que les deux autres méthodes. Il apparaît aussi que BOA est même moins rapide que l'algorithme génétique classique sur cette fonction 6-bipolar.

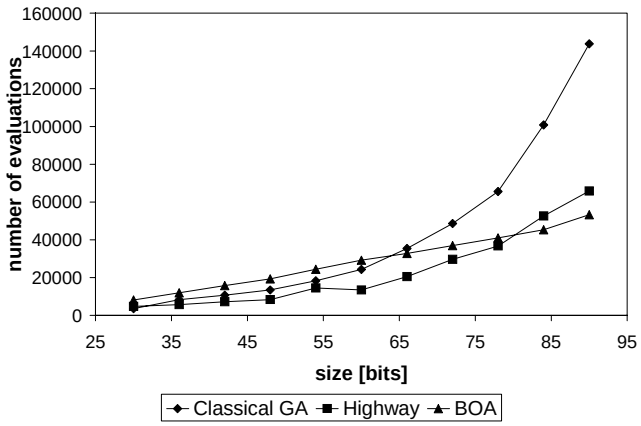


Figure 10. Nombre d'évaluations de la fonction de fitness pour la fonction 6-bipolar et pour les trois méthodes en moyenne sur 100 exécutions.

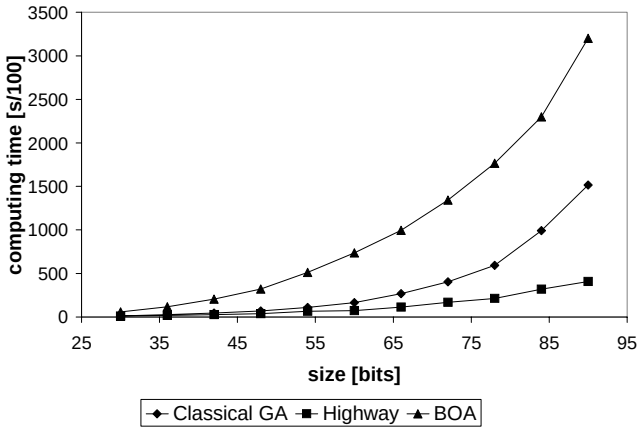


Figure 11. Temps de calcul total pour la fonction 6-bipolar et pour les trois méthodes en moyenne sur 100 exécutions.

Les deux méthodes d'adaptation présentées dans la section 2.3.2.3 ont été testées sur ces deux fonctions. Elles ont eu des performances très similaires. La méthode 1 marchait cependant légèrement mieux sur la fonction 6-bipolar et la méthode 2 sur la fonction 3-deceptive. Seuls les meilleurs résultats obtenus sont présentés ici. Les autres n'apportaient aucune modification aux commentaires précédents.

La table 1 montre le nombre de fois, parmi toutes les exécutions indépendantes (11 classes * 100 exécutions/classe = 1100 exécutions), où chacun des programmes échoue à découvrir la meilleure solution en 500 générations. Comme on peut le constater, c'est BOA qui a le plus d'échecs sur les deux fonctions (dans 7% des cas pour la fonction 6-bipolar). Les résultats entre l'algorithme génétique classique et Highway sont équivalents.

Missed solutions	**Classical GA**	**Highway**	**BOA**
3-deceptive	0	0	80
6-bipolar	0	2	24

Table 1. Nombre d'échecs de chaque programme pour découvrir le maximum de chaque fonction sur les 1100 exécutions indépendantes.

2.3.2.5 Conclusions

Les résultats présentés dans la section 2.3.2.4 montrent que Highway a de meilleures performances que l'algorithme génétique classique et que BOA en terme de temps de calcul. En fait, les performances pourraient être améliorées d'environ 10% par rapport à celles présentées du fait d'un nombre important d'opérations sur l'arbre de l'espace exploré utilisées uniquement à des fins d'analyse de l'exploration et donc

inutiles pour l'exploration elle-même. Ces résultats sont toutefois très préliminaires et ils ont été quelque peu étendus dans le travail présenté dans la section suivante.

On peut cependant déjà remarquer qu'une utilisation plus efficace des ressources de calcul disponibles était possible avec une analyse en temps réelle du comportement d'exploration de l'algorithme. On s'aperçoit également qu'une approche nécessitant la construction d'un modèle complexe, comme BOA, peut être fortement handicapée face à des méthodes plus simples lorsque le temps d'évaluation de la fonction de fitness est petit et qu'il est linéaire en fonction de la taille du problème. De plus, on peut constater que l'apport d'un modèle ne garantit pas une plus grande robustesse de la méthode au vu des mauvaises performances de BOA en terme de nombre d'échecs à la découverte du maximum. Ces constatations seront nuancées dans la section suivante où nous nous intéresserons à des problèmes plus difficiles pour lesquels la construction du modèle statistique sera un élément déterminant.

2.3.3 Comparaison de méthodes évolutives sur des problèmes avec dépendances

Nous avons mené récemment une étude sur l'effet des différents paramètres présentés dans la section 2.2.1 sur l'efficacité de plusieurs algorithmes d'optimisation combinatoire stochastique. Nous nous sommes intéressé au degré d'épistasie, à la contiguïté des variables et aux propriétés trompeuses ou non des fonctions à optimiser. Nous avons étudié le comportement d'un algorithme génétique classique et de l'algorithme BOA. Nous avons également utilisé plusieurs autres modèles statistiques simples de la population et analysé leurs performances sur ces mêmes problèmes. Les fonctions que nous avons utilisées pour réaliser ces tests sont toutes construites selon le principe de la somme de sous-fonctions indépendantes trompeuses ou non. Ainsi la fonction objectif F peut s'écrire :

$$F(X) = \sum_{i=1}^{m} F_i(X_i)$$

avec m le nombre de sous-fonctions F_i de taille $k = n / m$ et X_i les k variables de X impliquées dans la sous-fonction F_i. Les sous-fonctions F_i sont de deux types :

$$F_i(X_i) = \sum_{x_i \in X_i} x_i$$

si F_i n'est pas trompeuse et :

$$\begin{cases} F_i^t(X_i) = 1 - \dfrac{\left(\sum_{x_i \in X_i} x_i\right) + 1}{k} & si \sum_{x_i \in X_i} x_i < k \\ F_i^t(X_i) = 1 & si \sum_{x_i \in X_i} x_i = k \end{cases}$$

si F_i^t est trompeuse.

Le positionnement des variables de chaque sous-fonction dans la représentation complète d'une solution (un chromosome) peut se faire suivant deux types de configurations : la configuration contiguë et la configuration non-contiguë. Dans la première, la numérotation des variables de F_i ou de F_i^t se fait comme suit :

$$X_i = \{x_{(i-1)\cdot k+1}, x_{(i-1)\cdot k+2}, \ldots, x_{(i-1)\cdot k+k}\} \quad avec\ i \in \{1, \ldots, m\}$$

Dans la configuration non-contiguë, la numérotation des variables de F_i se fait comme suit :

$$X_i = \{x_i, x_{i+m}, \ldots, x_{i+(k-1)\cdot m}\} \quad avec\ i \in \{1, \ldots, m\}$$

Tous les tests ont été réalisés sur des processeurs Athlon 1.5 Ghz.

2.3.3.1 Algorithme génétique classique

L'algorithme génétique utilisé comporte une mutation ponctuelle de probabilité d'occurrence 0.05, un crossover à un point de probabilité d'occurrence 0.45 et un crossover à deux points de probabilité d'occurrence 0.5. Il utilise aussi une sélection par tournoi de taille 10. La première fonction testée est une fonction composée de 120 variables ($|X|$ =120) et de m = 120/k sous-fonctions F_i^t trompeuses de taille k. La valeur de k varie entre 4 et 12. On utilise la disposition des variables contiguës. La taille de la population de l'algorithme génétique utilisé est calculée à partir de la formule donnée par Pelikan et Goldberg, $Tp = 2^k \cdot n^{1.05}$, pour assurer la convergence de hBOA, cela de façon à pouvoir comparer les performances respectives des algorithmes dans les mêmes conditions. Pour chaque valeur de k, les résultats représentent le nombre d'exécutions, sur toutes les 60 effectuées à chaque fois, pour lesquelles le maximum a été atteint en moins de 100 générations. La table 2 donne le nombre d'échecs sur les 60 exécutions.

k	4	6	8	10	12
Nb échecs	58	7	0	0	0

Table 2. Nombre d'échecs de l'algorithme génétique classique parmi les 60 exécutions pour chaque valeur de k et pour la fonction trompeuse contiguë.

Les résultats obtenus en terme de temps de calcul et de nombre de générations pour découvrir le maximum sont donnés dans la figure 12. Du fait de l'importante mémoire nécessaire pour stocker les populations pour la valeur de k la plus élevée (Tp = 622 592 pour k = 12), nous avons limité la taille maximum de la population à Tp = 480 000. On peut constater que l'algorithme génétique classique n'a aucune difficulté pour

découvrir la bonne solution même dans des problèmes considérés comme très difficiles (n = 120, $k \geq 10$) dès que la population devient suffisamment grande. Il semble également que pour les hauts niveaux d'épistasie, la taille de la population nécessaire soit bien moins importante que celle demandée par hBOA. La taille de population de 2400 utilisée pour k = 4 est cependant trop faible pour découvrir le maximum dans la plupart des cas.

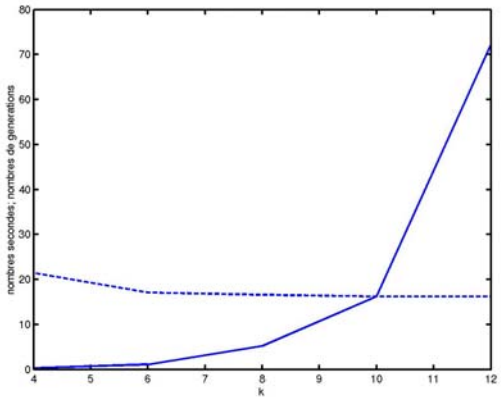


Figure 12. Temps moyen en secondes pour découvrir le maximum (ligne pleine) et nombre moyen de générations (ligne pointillée) en fonction de la taille k des sous-fonctions.

La deuxième fonction testée est la même que la première mais dans une configuration non-contiguë. Les résultats obtenus sont très différents de ceux de la première fonction. En effet, l'algorithme génétique classique est ici incapable de découvrir la solution (100% d'échecs) pour toutes les valeurs de k et cela même si la taille de la population dont il dispose est considérablement plus importante que celle demandée par hBOA. Par exemple, lorsque k est égal à 4 la taille de population devrait être de 2432 et nous avons utilisé une population de taille 480 000 ! On peut en déduire que les opérateurs utilisés pour explorer l'espace (la mutation et le crossover) sont inadaptés à la structure de ce problème. De fait, ils produisent un voisinage qui rend le paysage particulièrement dur à explorer. Les crossovers détruisent tous les blocs qui ont pu être formés précédemment à chaque application puisque les variables d'un même bloc vont obligatoirement être séparées par l'opérateur. Ainsi, deux points voisins dans ce paysage, c'est-à-dire distants d'une seule application d'un crossover, vont être totalement différents du point de vue de la fonction à optimiser. Par contre, dans la configuration de variables contiguës, les crossover deviennent des opérateurs de voisinage optimaux puisqu'ils vont conserver et mixer la majorité des blocs. Cela explique les performances extrêmement bonnes de l'algorithme génétique classique sur ce type de problèmes. On peut en conclure que dans le cas général, si le problème possède des dépendances et que celles-ci ne sont pas connues, la configuration utilisée a toutes les chances d'être non-contiguë et donc de rendre le problème impossible pour un algorithme génétique classique. Les problèmes que l'on peut résoudre avec un algorithme génétique classique sont donc ceux qui ne possèdent pas ou peu de dépendances ou ceux pour lesquels une

connaissance approfondie du problème permet de construire des opérateurs adaptés à la structure du problème.

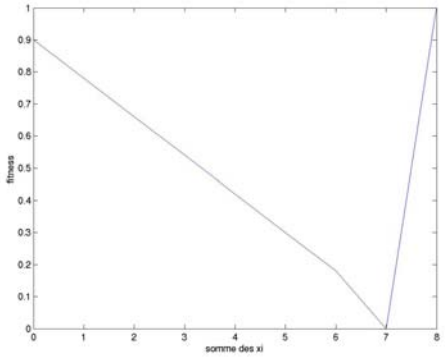
2.3.3.2 Algorithme hBOA

Le programme hBOA étant un produit commercial, nous n'avons eu accès qu'à la version de démonstration qui limite la taille des problèmes à 100 variables booléennes. La taille de population utilisée est calculée selon la formule $Tp = 2^k \cdot n^{1.05}$ et le nombre de génération maximum est de 100. Les autres paramètres sont les paramètres par défaut de hBOA. La première fonction testée est une fonction composée de 100 ou 96 variables et de $m = 100/k$ ou $m = 96/k$ (de façon à ce que m soit une valeur entière) sous-fonctions F_i^t trompeuses de taille k. La valeur de k varie entre 4 et 12 et la disposition des variables est contiguë. Les temps d'exécution de BOA étant considérablement plus élevés que pour l'algorithme génétique classique, les résultats présentés sont des moyennes réalisées uniquement sur 6 exécutions. Les résultats étant beaucoup plus variables qu'avec l'algorithme génétique classique, chaque situation sera détaillée indépendamment.

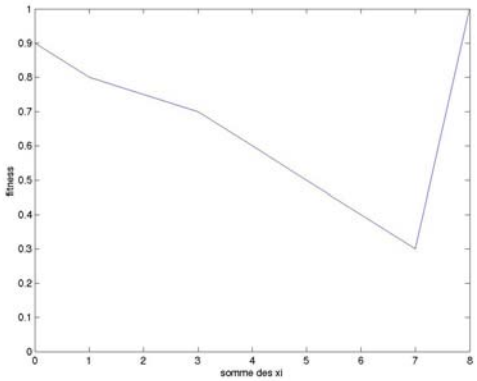
- $n = 100$, $k = 4$, $Tp = 2\ 016$. Le maximum global (25) est atteint lors de chaque exécution avec un nombre moyen de génération de 25 et un temps moyen d'exécution de 20 secondes.
- $n = 96$, $k = 6$, $Tp = 7\ 720$. Le maximum global (16) n'a jamais été atteint lors des 6 exécutions. La valeur moyenne atteinte de 15.7 étant très supérieure à la valeur "trappe" de 14.4 et proche du maximum global, on peut raisonnablement supposer que ce maximum aurait été atteint avec un petit nombre de générations supplémentaires. Le temps moyen d'exécution était de 13 minutes.
- $n = 96$, $k = 8$, $Tp = 30\ 720$. Le maximum global (12) n'a jamais été atteint lors des 6 exécutions. La valeur moyenne atteinte de 10.8 étant la valeur trappe dans tous les cas, on peut supposer que le processus était bloqué durablement dans le maximum local et ne pouvait pas atteindre le maximum global. Le temps moyen d'exécution était de 1 heure 53 minutes. Une deuxième expérimentation a été menée consistant à doubler la taille de la population ($Tp = 61\ 440$) pour ce même problème. Le temps moyen d'exécution est passé à 4 heures 50. Le maximum global n'a jamais été atteint dans les 3 exécutions réalisées et la valeur trappe légèrement dépassée avec une valeur de 10.9 à chaque fois.
- $n = 100$, $k = 10$, $Tp = 129\ 000$. Le maximum global (10) n'a jamais été atteint lors des 6 exécutions. La valeur moyenne atteinte de 9.1 étant la valeur trappe dans tous les cas, on peut supposer que le processus était bloqué durablement dans le maximum local et ne pouvait pas atteindre le maximum global. Le temps moyen d'exécution était de 12 heures 40 minutes.
- $n = 96$, $k = 12$, $Tp = 491\ 520$. Le temps d'exécution moyen étant de 61 heures, nous n'avons pu lancer que 2 exécutions. Le maximum global (8) n'a jamais été atteint lors des 2 exécutions. La valeur moyenne atteinte de 7.4 étant légèrement supérieure à la valeur trappe de 7.2, on peut supposer que le processus n'était pas définitivement bloqué et que le résultat pouvait encore être amélioré.

On peut d'ores et déjà constater que, contrairement à ce qui à été publié, hBOA n'est pas capable, dans toutes les situations pour lesquelles il est censé fonctionner correctement, de découvrir la solution dans les conditions pour lesquelles des preuves de convergences ont été données. Les résultats sur hBOA présentés dans la littérature ont toujours été faits sur des fonctions possédant peu de dépendances (maximum 6 ou 8). Le temps d'exécution devient prohibitif dès que la taille des dépendances augmente tout en ne garantissant pas de découvrir le maximum, ni même d'échapper aux maximums locaux trappes. Il semble que les heuristiques utilisées au cours de la construction du modèle bayésien et qui ne sont pas prises en compte dans les preuves de convergence aient des conséquences très importantes sur la qualité de l'exploration effectuée.

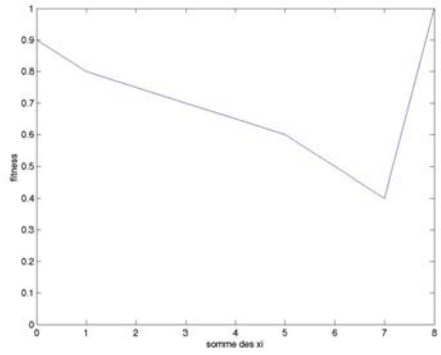
Au cours de ces tests nous avons remarqué une sensibilité particulière de l'algorithme hBOA à la forme de la fonction à optimiser. Ce phénomène n'étant décrit dans aucune publication à notre connaissance et n'apparaissant pas dans les calculs de convergences fournis par Pelikan et Goldberg, nous avons réalisé une autre série de tests dans le but de mieux cerner le phénomène. La fonction F_i^t testée dans le problème n = 96, k = 8, Tp = 30 720 était la fonction suivante :



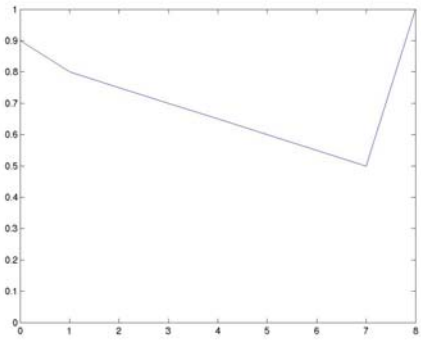
En changeant cette fonction, mais en restant dans les contraintes n = 96, k = 8, Tp = 30 720, on obtient des résultats très différents. Pour la fonction :



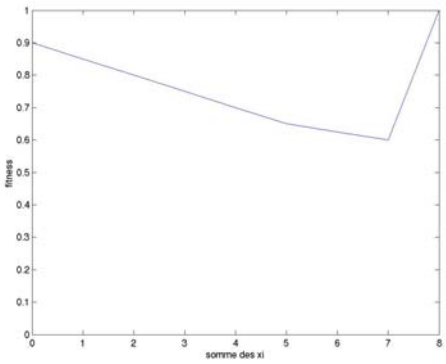
on obtient les mêmes résultats, c'est-à-dire le maximum global jamais atteint et le processus reste bloqué sur la valeur trappe 10.8. Pour la fonction :



on obtient deux maximums globaux sur six exécutions et une moyenne de 54 générations. Pour les quatre exécutions ayant échoué le score moyen est de 11.85, c'est-à-dire très proche du maximum global. Pour la fonction :



on obtient cinq maximums globaux sur six exécutions et une moyenne de 15 générations. Pour l'exécution ayant échoué le score est de 11.9, c'est-à-dire très proche du maximum global. Pour la fonction :



on obtient le maximum global pour les six exécutions et une moyenne de 16 générations.

Bien que ces résultats mériteraient d'être confirmés par un plus grand nombre d'expérimentations le phénomène apparaît relativement clairement. Il semble qu'un écart trop important entre la valeur de la solution la plus similaire à la solution maximum global et cette valeur de maximum global rende la construction du modèle statistique difficile et ne permette pas de détecter les bonnes dépendances. Bien que l'heuristique de construction du réseau bayésien de hBOA n'ait pas été décrite dans le détail, au vu de ces résultats on peut supposer qu'elle repose, au moins en partie, sur les fréquences univariées. En effet, la succession de fonctions ci-dessus tend à privilégier de plus en plus la sélection de vecteurs contenant des 1. Or le maximum global est constitué entièrement de 1. Les fonctions permettant la sélection des 1 favoriseraient donc la construction d'un modèle prédisant des 1 à toutes les positions. Une autre piste pour expliquer ce phénomène peut être liée au système utilisé dans hBOA pour générer la population à partir du modèle. Comme nous l'avons souligné dans la section 2.2.2.2, l'algorithme donné par Pelikan et Goldberg pour la génération de la nouvelle population n'est pas clair puisqu'il n'indique pas comment se fait l'initialisation. Il est alors possible que cette initialisation se fasse à partir des informations de fréquences univariées, biaisant donc la population en faveur des valeurs instanciées à la génération précédente.

Toutes ces expérimentations ont été réalisées en configuration des variables contiguë. Nous avons effectué les mêmes expérimentations en configuration non contiguë et les résultats ont été pratiquement identiques à ceux obtenus en configuration contiguë. Cela signifie que l'approche de hBOA pour la construction du modèle statistique et la génération de la nouvelle population n'est pas perturbée par la position des variables participant à un même bloc. C'est ce qui était attendu puisque rien dans l'algorithme de hBOA ne repose sur la position des variables dans le vecteur chromosome, cette position ne doit donc avoir aucune influence sur l'efficacité de la méthode. C'est un résultat important car, comme nous l'avions vu dans la section précédente, l'algorithme génétique classique est incapable, dans cette situation, de découvrir les blocs pour échapper aux maximums locaux trappes. hBOA est donc bien capable de découvrir et d'exploiter les dépendances d'un problème dans une situation difficile pour peu que la taille de ces dépendances reste raisonnable et que la structure de la fonction n'empêche pas les heuristiques de construction du modèle et de génération de la population de guider l'exploration dans la bonne direction.

2.3.3.3 Autres modèles statistiques.

Nous avons étudié d'autres modèles statistiques pour découvrir les dépendances et concu des opérateurs basés sur ces modèles pour générer de nouvelles solutions. Nous avons voulu vérifier ce qu'il était possible de découvrir sans chercher de dépendances de grande taille. Nous avons limité nos modèles aux mesures des dépendances de tailles deux et observés ce qu'il était possible de faire avec de tels modèles sur des problèmes possédant des dépendance de plus grande taille (ceux présentés dans les sections 2.3.3.1 et 2.3.3.2). Nous avons utilisé un algorithme génétique basé sur la sélection par tournois. A chaque génération nous évaluons la fréquence de tous les couples d'instanciations des

variables rencontrés dans le population. Nous utilisons ensuite ces statistiques pour calculer des mesures de corrélations entre les instanciations des variables. Nous définisons des opérateurs génétiques spécifiques utilisant ces mesures pour produire de nouvelles solutions afin de constituer la nouvelle population.

Nous avons comparé différentes mesures du point de vu de leur capacité à découvrir les couples de variables dépendantes au sein des blocs de taille supérieure à deux (puisque dans les problèmes de tests la taille des blocs varie de 4 à 12). Nous avons utilisé les mesures suivantes :

- Les fréquences conjointes des instanciations de deux variables :

$$f(x_i = \phi_1, x_j = \phi_2)\ \forall i, j \in \{1,\cdots,n\}\ et\ \phi_1, \phi_2 \in \{0,1\}$$

- La probabilité conditionnelle:

$$P(x_i = \phi_1 \mid x_j)\ \forall i, j \in \{1,\ldots,n\}\ et\ \phi_1 \in \{0,1\}$$

- L'information mutuelle $M(x_i, x_j)$:

$$M(x_i, x_j) = \sum_{x_i=\phi_1, x_j=\phi_2} f_{x=\phi_i x_j=\phi_2} \cdot \log \frac{f_{x_i=\phi x_j=\phi_2}}{f_{x_i=\phi_1} \cdot f_{x_j=\phi_2}} \quad \forall i, j \in \{1,\cdots,n\}\ et\ \phi_1, \phi_2 \in \{0,1\}$$

- L'implication statistique (Blanchard J. et al., 2003; Kuntz P. et al., 2002) $\varphi\left(x_i = \phi_1, x_j = \phi_2\right)$:

$$q(x_i = \phi_1, x_j = \overline{\phi_2}) = \frac{n_{x_i=\phi_1 \cap x_j=\overline{\phi_2}} - \frac{n_{x_i=\phi_1} \times n_{x_j=\overline{\phi_2}}}{Tp}}{\sqrt{\frac{n_{x_i=\phi_1} \times n_{x_j=\overline{\phi_2}}}{Tp}}} \quad \forall i, j \in \{1,\cdots,n\}\ et\ \phi_1, \phi_2 \in \{0,1\}$$

avec $n_{x_i=\phi_1}$ et $n_{x_j=\overline{\phi_2}}$ le nombre de fois où x_i est instanciée à ϕ_1 et x_j est instanciée à non ϕ_2 respectivement dans tout l'échantillon de taille Tp et :

$$\varphi\left(x_i = \phi_1, x_j = \phi_2\right) = \frac{1}{\sqrt{2\pi}} \int_{q(x_i=\phi_1, x_j=\overline{\phi_2})}^{\infty} e^{-\frac{t^2}{2}} dt$$

Cette dernière mesure est un indice du degré d'implication entre l'instanciation de deux variables. Elle a été montrée comme étant beaucoup plus sensible que la probabilité conditionnelle.

Nous avons pu constater qu'aucune de ces quatre mesures n'est capable de détecter les instanciations de deux variables correspondant à l'instanciation du maximum global du bloc auquel elles appartiennent. Par exemple, dans le cas d'un problème avec des blocs de taille 8, si les variables x_1 à x_8 appartiennent à un même bloc et qu'elles doivent être instanciées à 1 pour que la fonction objectif atteigne son maximum global et à 0 pour atteindre le maximum local, les quatre mesures détectent les dépendances les plus fortes pour les couples de variables instanciées à 00 et les plus faibles pour les couples de variables instanciées à 11. Nous confirmons ainsi qu'il semble impossible de détecter les bonnes dépendances si on ne considère pas ensemble toutes les variables d'un bloc (dans l'exemple, il faudrait mesurer les dépendances entre les 8 variables simultanément). Lorsque les dépendances ne sont pas connues, il est donc bien nécessaire de les déterminer complètement pour avoir une chance de résoudre le problème. Le nombre de combinaisons correspondant au nombre de dépendances possibles est le nombre de partitions possibles des *n* variables et est donc exponentiel avec *n*. Il faut donc soit utiliser une heuristique, par exemple celle utilisée dans hBOA pour construire le réseau bayésien, soit utiliser des informations connues sur la structure du problème pour déterminer ces dépendances et pouvoir en déduire la solution optimale (par exemple en triant les variables pour se placer dans une configuration contiguë).

Dans cette optique, nous avons choisi d'utiliser une information sur la structure du problème, le fait que les sous-fonctions sont trompeuses, pour essayer de construire un algorithme performant pour résoudre ces problèmes. Nous utilisons un algorithme génétique de type PMBGA, basé sur les quatre mesures définies précédemment. Nous avons construit un opérateur de génération de nouvelles solutions extrêmement simple. Etant donné que les sous-fonctions que nous essayons d'optimiser sont trompeuses, nous avons choisi de privilégier les instanciations qui sont les moins fréquentes dans la population après la sélection par tournois. En effet, avec des sous-fonctions trompeuses, les instanciations de blocs correspondant aux maximums locaux ou similaires, du point de vue de la distance de Hamming, aux maximums locaux ont des probabilités beaucoup plus importantes d'être présentes dans la population et de constituer les meilleures solutions. Ces instanciations sont donc préférentiellement sélectionnées et par conséquent les instanciations correspondant aux maximums globaux sont très largement sous représentées dans la population échantillon. Notre opérateur consiste donc à sélectionner aléatoirement deux variables x_i et x_j , puis de choisir parmi les quatre instanciations possibles de ces deux variables celle, $\phi_i\phi_j$, qui à la valeur de mesure la plus faible (correspondant donc à la corrélation la moins forte). Puis, en prenant l'instanciation ϕ_i pour la variable x_i, nous choisissons pour toutes les autres variables x_k l'instanciation de la façon suivante :

$$choix(x_k) = \arg\min_{\phi_k} mesure(x_i = \phi_i, x_k = \phi_k) \qquad \forall k \neq i$$

avec $mesure(x_i = \phi_i, x_k = \phi_k)$ l'une des quatre mesures définies plus haut. Nous utilisons les instanciations ainsi choisies pour construire une nouvelle solution. Nous avons utilisé cet algorithme pour tous les problèmes testés dans la section 2.3.3.2. Nous avons pu constater, en utilisant une population de la même taille que celle utilisée par hBOA[17], que nous trouvons la bonne solution en une seule génération (c'est-à-dire le calcul de la première population aléatoire, la sélection par tournois, le calcul d'une des quatre mesures et finalement la construction d'une solution) et pour tous les problèmes, même ceux de tailles 120 avec des blocs de taille 12. Le temps de calcul correspond au temps nécessaire pour évaluer la fitness de toute la population et pour le calcul des mesures à partir des statistiques des couples dans la population sélectionnée. Nous trouvons donc la bonne solution en quelques minutes sur des problèmes pour lesquels hBOA ne trouvait pas la solution après plusieurs jours de calculs! Ainsi, même sans connaître les dépendances ni leur nombre à l'avance, mais en utilisant l'information que la fonction à optimiser est trompeuse, nous pouvant découvrir la bonne solution en un temps très court et cela même pour des problèmes difficiles.

Pour déterminer les limites de notre approche, nous avons construit un problème constitué de sous-fonctions pour moitié trompeuses et pour moitié non trompeuses. Dans ce cas notre approche est incapable de découvrir la bonne solution. Nous avons modifié notre méthode pour rendre la génération de nouvelles solutions plus stochastiques, avec une probabilité de construire une nouvelle solution à partir d'opérateurs standards de type crossover ou mutation, avec l'opérateur défini précédemment et avec des opérateurs faisant des choix d'instanciations mixtes entre les valeurs des mesures statistiques élevées ou basses (c'est-à-dire choisir les instanciations spécialement corrélées ou spécialement peu corrélées). Même avec ces nouveaux opérateurs notre approche échoue à découvrir la solution optimale. Cela s'explique facilement par le fait que de devoir découvrir quelles variables participent à des sous-fonctions trompeuses et quelles variables participent à des sous-fonctions non trompeuses est lui-même un problème combinatoire NP-complet. Nous avons pu vérifier que ce type de problème mixte ne perturbe pas hBOA. En effet, hBOA à obtenu les mêmes performances avec ce type de fonctions que celles qu'il a obtenues dans la section 2.3.3.1.

2.4 Conclusion

Il existe de nombreuses propriétés entrant en jeu dans la complexité intrinsèque d'un problème d'optimisation combinatoire global. La complexité effective dépend également en bonne partie de la stratégie d'exploration utilisée. On peut distinguer deux grandes classes de stratégies : (1) celles basées sur des grimpeurs exploitant la topologie du paysage formé par l'opérateur de voisinage utilisé pour explorer l'expace; (2) celles basées sur la découverte de la structure de la fonction à optimiser, par modélisation statistique d'un échantillon biaisé de l'espace de recherche, pour déterminer un voisinage intéressant pour l'exploration. Selon la nature du problème, l'une ou l'autre de ces classes

[17] Nous avons également pu constater que ces résultats restaient vrais avec des tailles de population de 2 à 4 fois inférieure.

de stratégies peut s'avérer efficace. Si la première classe est handicapée lorsque le voisinage choisi (le plus souvent le voisinage de Hamming) produit un paysage fortement multimodal et/ou trompeur, la deuxième est en difficulté si le nombre de dépendances entre les variables du problème est élevé. Nous avons montré en particulier les limitations de la meilleure approche existant dans cette deuxième classe, hBOA, sur des problèmes où le nombre de dépendances n'est pas négligeable.

Les problèmes réels, en particulier ceux existant en bioinformatique dont nous verrons quelques exemples dans le chapitre suivant, sont souvent difficiles car ils possèdent à la fois beaucoup de dépendances non connues et produisent un paysage fortement multimodal pour un voisinage standard. Il est alors important de bien comprendre les possibilités de chacune des approches sur les différents types de difficultés rencontrées pour adapter sa stratégie d'exploration au problème traité. C'est ce que nous avons essayé de faire, ou tout du moins de commencer à faire, sur un certain nombre de problèmes bioinformatiques réels et difficiles. Nous présentons dans le chapitre suivant plusieurs de ces problèmes avec à chaque fois une stratégie d'exploration de l'espace du problème dédiée et combinant plusieurs méthodes appartenant aussi bien à la classe des méthodes par grimpeurs qu'à celle des méthodes par découverte de structure. Dans tous les cas, c'est l'expertise sur le problème donné qui a été la clé pour la conception d'une stratégie efficace.

CHAPITRE 3

Applications bioinformatiques

Depuis le début des années 90, la biologie et plus particulièrement la biologie moléculaire et la génétique ont connu de grands bouleversements. De nouvelles technologies sont apparues, comme le séquençage, la PCR (Polymerase chain reaction) ou la spectrométrie de masse, fournissant des quantités de données phénoménales en un temps extrêmement court. Ces données ont apporté une nouvelle vision sur beaucoup de mécanismes biologiques et ont nécessité le développement de méthodes d'analyses inédites. L'informatique est alors devenue indispensable pour la conception de ces méthodes et leurs applications à la grande quantité de données disponible et a donné naissance à une nouvelle science : la bioinformatique.

Beaucoup des problèmes de bioinformatique impliquent une quantité de calculs extrêmement importante. Le développement d'algorithmes spécifiques performants est donc une nécessité. Nombre de ces problèmes peuvent être transcrits en des problèmes d'optimisation combinatoire et donc profiter des performances des méthodes d'optimisation combinatoires pour être résolus en un temps raisonnable (Greenberg H.J. et al., 2004). Depuis quelques années, les métaheuristiques ont été employées avec succès sur un grand nombre de problèmes bioinformatiques (Fogel G.B. and Corne D.W., 2003; Gras R. et al., 2004). Les sujets abordés sont très divers : l'alignement multiple de séquences global (Notredame C. et al., 1997) ou local (Fogel G.B. et al., 2004; Hernandez D. et al., 2004), la prédiction de fonction de protéines (Frey J. et al., 2004), l'alignement de structure tridimensionnelle de protéines (protein threading) (Lathrop R.H. et al., 2001; Yanev N. and Andonov R., 2003), l'inférence de réseau d'interaction de protéines (Kimura S. et al., 2003; Kimura S. et al., 2004), la sélection de marqueurs biologiques (Petricoin III E.F. et al., 2002), l'identification des protéines par spectrométrie de masse (Gras R. et al., 1999), l'analyse de données de puces à ADN (Jourdan L. et al., 2004)…

Je présente dans ce chapitre différents travaux en bioinformatique que j'ai réalisés ou dirigés depuis ma soutenance de thèse. Ils portent sur deux grands sujets : l'identification automatique des protéines en utilisant des données de spectrométrie de masse et l'alignement multiple local de séquences biologiques. Presque tous ces travaux reposent, au moins en partie, sur des approches métaheuristiques.

3.1 Identification des protéines

3.1.1 Principes

La protéomique a pour but d'étudier l'expression de l'ensemble des protéines d'un organisme ou d'un tissu à un moment particulier dans des conditions particulières (Bennington S.R. and Dunn M.J., 2001; Wilkins M.R. et al., 1997). Ainsi le niveau d'expression de chaque protéine peut être corrélé avec des situations que l'on souhaiterait pouvoir prédire, par exemple des maladies. La méthodologie généralement suivie et présentée dans la figure 13 pour réaliser ce genre d'étude se décompose en : (1) prélever un échantillon de tissu dans les deux populations à comparer, par exemple une population d'individus sains et une population d'individus porteurs d'une maladie; (2) séparer les protéines de l'échantillon par des méthodes de 2D SDS-PAGE (Klose J., 2002) ou D'HPLC (Nelson R.W. et al., 2000) dans lesquelles on fait migrer les protéines à différentes positions dans un gel ou un liquide en fonction de leur masse, de leur pH,… ; (3) digérer les protéines par un ou plusieurs enzymes pour obtenir des ensembles de peptides; (4) mesurer la masse de ces peptides ou de fragments de ces peptides par spectrométrie de masse; (5) identifier les protéines correspondantes par comparaison de ces masses avec celles provenant de base de données de séquences protéiques digérées virtuellement. C'est ce dernier point qui sera étudié plus en détail dans les sections suivantes.

La première technique utilisée pour l'identification des protéines fut le séquençage d'Edman basé sur la chromatographie (Edman P. and Begg G., 1967). Cette méthode bien que fiable et totalement automatisable était longue et coûteuse (environ un jour par protéine) et inapplicable en cas de modification sur la protéine à identifier. Au début des années 90, une nouvelle technique a vu le jour basé sur la spectrométrie de masse (Fenyo, 2000; Gras R. et al., 2003b; Gras R. and Muller M., 2001). Son principe est le suivant : mesurer la masse d'une série de fragments caractéristiques d'une protéine et comparer les masses obtenues avec des masses théoriques calculées à partir des séquences protéiques contenues dans les bases de données. Un score de ressemblance entre les masses expérimentales et les masses théoriques est calculé permettant de déterminer quelle protéine de la base de données est la plus susceptible de correspondre à la protéine expérimentale. Il existe deux types d'approches différentes d'identification par spectrométrie de masse liés à deux techniques de spectrométrie différentes : l'identification par *empreinte de masse peptidique* (peptide mass figerprint, PMF) et l'identification par *spectrométrie de masse en tandem* (tandem mass spectrométrie, MS/MS). Ces deux techniques nécessitent un prétraitement des spectres appelé détection de pics (Gras R. et al., 1999). Il consiste à extraire des spectres bruts, produits par les spectromètres et mesurant la fréquence d'observation des valeurs de masses/charges issues de l'échantillon, les valeurs de masses correspondant réellement aux fragments protéiques de l'échantillon. Cette étape est sujette à des problèmes de calibration du spectromètre, de chevauchement de plusieurs fragments pour une même valeur de masse/charge et de détection de la charge (suivant le type de spectromètre utilisé). Les

erreurs éventuelles pouvant intervenir dans cette étape doivent être prises en considération lors de l'identification proprement dite.

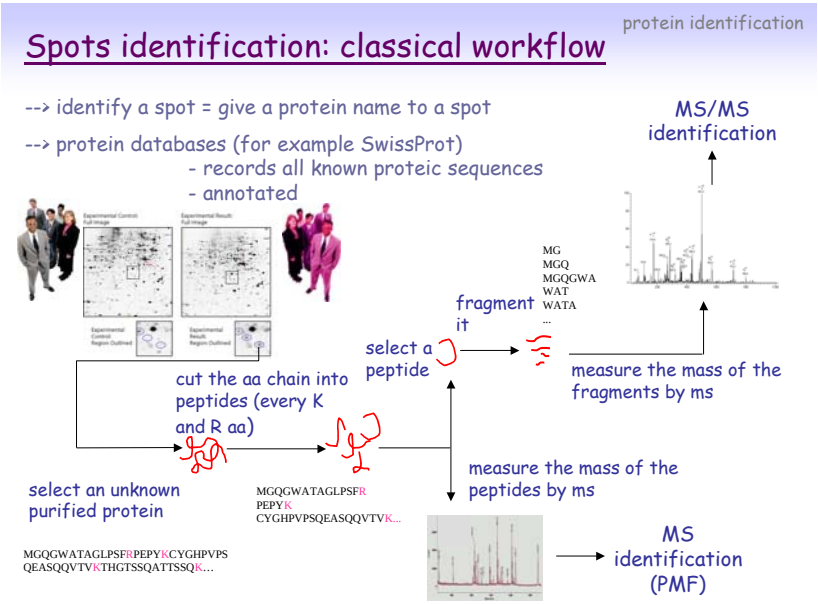


Figure 13. Le processus classique de l'analyse protéomique

3.1.2 Identification par empreinte de masse peptidique

3.1.2.1 Etat de l'art

L'identification par empreinte de masse peptidique a été inventée simultanément par plusieurs groupes en 1993 (James et al., 1993; Karlin S. and Altschul S.F., 1993; Mann M et al., 1993; Pappin D.D.J. et al., 1993; Yates et al., 1993). Il existe maintenant de nombreux outils basés sur le PMF qui procèdent de façon similaire et diffèrent principalement par la fonction de score choisie pour mesurer la ressemblance entre les masses expérimentales et les masses théoriques. Leur principe est le suivant : produire pour chaque séquence protéique (ou séquence génomique traduite en séquence d'acides aminés) d'une base de données, une liste de masses théoriques des peptides formés par une digestion virtuelle (souvent par la trypsine) de la séquence puis de comparer ces listes à la liste de masses expérimentales issues de la détection de pic. La comparaison consiste à apparier les masses expérimentales avec les masses théoriques de chaque protéine de la base de données si la différence entre une masse théorique et une masse expérimentale est inférieure à un seuil d'erreur fixé. Enfin, pour chaque liste de masses expérimentales appariées, donner un score qui va mesurer la ressemblance entre la protéine de la base de données et la protéine de l'échantillon. Il existe de nombreux facteurs qui vont contribuer à rendre cette comparaison difficile et perturber la qualité de la mesure de similarité. Le processus expérimental de digestion n'est pas parfait et peut mener à ce que l'on appelle un *miss cleavaged*, c'est-à-dire deux peptides consécutifs qui ne sont pas séparés par l'enzyme de coupure et dont la masse totale est mesurée. Le spectromètre peut être mal

calibré et produire des listes de masses de valeurs décalées. La précision du spectromètre peut être variable en fonction de la masse mesurée. Certains peptides peuvent ne pas être mesurés par le spectromètre du fait de propriété physico-chimique empêchant qu'ils soient chargés correctement par exemple. De nombreux contaminents peuvent être mesurés : des molécules utilisées dans le processus de séparation des protéines ou dans le processus de préparation à la mesure par le spectromètre, des peptides provenant de l'environnement (kératine humaine par exemple), protéines mal séparées et donc mélangées dans l'échantillon… Les protéines peuvent être modifiées soit par des adjonctions chimiques (modifications chimiques) lors de la préparation de l'échantillon soit naturellement dans l'organisme (modifications post-traductionnelles)[18]. Ces modifications vont changer la masse des peptides, ce dont il faudra tenir compte lors de la comparaison avec les masses théoriques. Enfin, du bruit lors de la mesure par le spectromètre peut être interprété comme une masse de fragment lors de la détection de pic et être appariée à tord avec une masse théorique. La qualité de l'identification effectuée sera donc fortement dépendante de la capacité de la mesure de similarité à prendre en compte ces différents paramètres de manière pertinente.

La manière la plus simple d'évaluer cette similarité est de compter le nombre d'appariements effectués entre la liste expérimentale et la liste théorique. La plupart des premières méthodes d'identification fonctionnaient avec ce type de score. PeptIdent (Wilkins M.R. et al., 1999) utilise aussi ce score mais il incorpore dans la construction de la liste de peptides théoriques des informations importantes provenant des annotations sur les mutations, les modifications post-traductionnelles et les variants d'*épissages* (splice variants) provenant de la base de données Swiss-Prot (Boeckmann B. et al., 2003). En plus du peu de paramètres pris en compte par ce score il a l'inconvénient majeur d'être biaisé en faveur des grandes protéines. En effet, les grandes protéines ont plus de peptides que les autres et elles ont donc une plus grande probabilité d'avoir un nombre important d'appariements entre leurs masses théoriques et les masses expérimentales. Le score de Mowse (Pappin D.D.J. et al., 1993) cherche à limiter cet effet en considérant le pourcentage de couverture de la séquence par les peptides appariés. Il utilise de plus la propriété de la distribution des masses de peptides dans les bases de données pour pondérer plus fortement les appariements avec des peptides rares. Mascot (Perkins D.N. et al., 1999) améliore le score de Mowse en calculant la probabilité d'apparier les masses expérimentales avec les masses théoriques par chance (malheureusement, Mascot étant un produit commercial, son score n'a jamais été publié explicitement). Plus cette probabilité est basse et plus le score est significatif. Un seuil de significativité S de cette probabilité est donné qui dépend du nombre N de protéines dans la base de données de telle façon que $N \cdot S << 1$ et donc que l'on ne s'attende pas pour cette taille de base de données à obtenir un appariement avec cette valeur de score par hasard. Profond (Zhang and Chait, 2000) utilise les probabilités bayésiennes pour calculer la probabilité de l'appariement entre les masses expérimentales et théoriques. Ce modèle permet de réduire considérablement le biais envers les grandes protéines et de prendre en compte

[18] Ce dernier phénomène est particulièrement important puisque les processus biologiques impliquant des protéines sont très dynamiques et la fonction d'une protéine peut changer par l'adjonction de ces modifications. Cela peut être lié à la maladie dont on cherche à déterminer les causes et il est donc important de pouvoir détecter ces modifications.

des informations diverses comme des données sur les conditions expérimentales ou l'erreur (différence de masse) réalisée lors des appariements. Les travaux récents portent sur l'incorporation dans le modèle de score d'un maximum de paramètres pouvant participer à une bonne discrimination de la protéine dans la base de données correspondant à celle de l'échantillon. Un papier publié récemment (Magnin J. et al., 2004) propose un score basé sur un test d'hypothèse par ratio de probabilité. Les probabilités d'un appariement par hasard et d'un vrai appariement, H_0 et H_1 respectivement, sont calculées pour une liste de masses donnée. Les modèles utilisés pour H_0 et H_1 incorporent, entre autres, la notion de couverture de la séquence et de la fréquence des acides aminés et des modifications. Ces paramètres sont considérés comme statistiquement indépendants et sont estimés par apprentissage sur des vraies identifications pour H_1 et sur des appariements aléatoires pour H_0. Les auteurs montrent de bonnes performances pour leur approche en comparaison avec d'autres méthodes existantes.

3.1.2.2 SmartIdent

Le premier travail que j'ai mené dans le domaine de la protéomique a consisté a développer d'un outil d'identification PMF, SmartIdent (Gras R. et al., 1999; Gras R. et al., 2000), ayant pour but de fonctionner autant que possible en mode automatique. C'est le premier outil de PMF qui ait été réalisé en suivant une approche par apprentissage et il m'a permis de me familiariser avec les concepts d'algorithmes évolutifs. Constatant que les méthodes de PMF existantes proposaient soit des scores extrêmement simples soit des scores probabilistes qui ne prenaient en considération qu'une petite partie des paramètres influant sur l'identification, l'idée a été de construire un score heuristique prenant en compte un maximum de facteurs et d'optimiser ce score par apprentissage des coefficients de pondérations déterminant l'importance des différents facteurs. Le score de SmartIdent est divisé en deux parties, une première partie (le score de peptide) mesure la vraisemblance d'un appariement entre un peptide théorique et une masse expérimentale et la deuxième (le score de protéine) évalue la cohérence des informations décrivant la protéine dans son ensemble avec les données expérimentales correspondant à un spectre complet. Les paramètres inclus dans le score de peptide sont le nombre de miss cleavaged, le nombre de modifications, l'intensité du signal du pic expérimental et la valeur d'hydrophobicité (échelle GRAVY (Kyte J. and Doolittle R.F., 1982)) du peptide théorique. Ceux inclus dans le score de protéine sont la déviation standard des erreurs d'appariement masses, le pourcentage de couverture de la protéine par les peptides appariés ainsi que les déviations avec les valeurs expérimentales de point isoélectrique et de masse moléculaire (déterminées à partir du gel d'électrophorèse bidimensionnelle utilisé pour la séparation des protéines). Le score final d'une protéine de la base de données est donné par :

$$score = \left(\sum_{peptides\ apparies} score\,de\ peptide \right)^3 \cdot (score\,de\ proteine)$$

Les pondérations des paramètres dans les deux sous-scores ont été apprises par un algorithme génétique à partir d'un ensemble de 174 spectres pour lesquels une identification sûre existait. L'algorithme génétique utilisé optimisait des valeurs réelles selon le modèle présenté dans (Michalewicz Z., 1996). Il était de plus basé sur un modèle en îlots à deux populations (Cantu-Paz E. and Goldberg D.E., 1999). Le principe d'utilisation de ces deux sous-populations était les prémices de ce que nous avons utilisé ultérieurement dans notre algorithme PyM (Frey J. et al., 2004). Les paramètres de l'algorithme génétique pour une des deux sous-populations permettait une exploration rapide de l'espace de recherche (grande taille, forte probabilité de mutation et de crossover) alors que l'autre sous-population était dédiée à l'exploitation (petite taille, faible probabilité de mutation). Les communications entre sous-populations étaient unidirectionnelles, de la sous-population exploratrice vers la sous-population chargée de l'exploitation. La fonction *F* à maximiser avait pour but de découvrir un score maximalement discriminant, de telle façon que le score obtenu par la protéine réellement dans l'échantillon soit le plus élevé parmi toutes les protéines de la base de données et le plus éloigné de celui obtenu par la protéine obtenant le deuxième meilleur score :

$$F(X) = \begin{cases} \dfrac{score_0}{score_0 + score_1} & si\ la\ bonne\ proteine\ est\ en\ premiere\ position \\ 0.5 - (rang * 0.05) & si\ la\ bonne\ proteine\ est\ dans\ les\ 10\ premieres \end{cases}$$

avec $score_0$ le score le plus élevé pour toutes les protéines de la base de données, $score_1$ le deuxième meilleur score et *rang* le rang de la bonne protéine dans la liste des scores. L'apprentissage a permis de découvrir une fonction de score particulièrement robuste et fiable pour l'identification par PMF.

Un avantage supplémentaire de cet outil est sa manière de traiter le problème de la calibration et de prendre en compte l'erreur de masse dans les appariements. Pour chaque protéine de la base de données, une régression linéaire robuste est réalisée entre les masses théoriques et expérimentales appariées. Cette régression permet de corriger les problèmes de calibrations au cours de l'identification en utilisant la cohérence des appariements réalisés. De plus, les appariements situés trop loin de la droite de régression sont éliminés. Cette étape associée à la calibration par régression permet d'être très tolérant dans les erreurs acceptées pour les appariements puisque ces erreurs seront corrigées par la suite. Finalement, la valeur de déviation standard à la droite régression est l'un des paramètres utilisés dans le score apportant un jugement sur la qualité globale de l'appariement entre les masses théoriques et expérimentales. Ainsi SmartIdent est capable de faire des identifications correctes même lorsque les données sont très mal calibrées. Cette approche innovante a été reprise dans plusieurs autres outils d'identification par PMF (Egelhofer et al., 2002; Parker K.C., 2002).

Une limitation de cette approche est sa dépendance aux données d'apprentissage. La quantité et la diversité des spectres disponibles n'étaient pas suffisantes d'une part pour couvrir toutes les situations expérimentales et les différentes qualités de spectres possibles et d'autre part pour déterminer les valeurs optimales du grand nombre de paramètres utilisés dans la fonction de score (plus de 30). De plus, les choix effectués

dans la structure de la fonction de score étaient empiriques et beaucoup d'autres fonctions auraient pu être essayées. L'approche que nous avons suivie pour Popitam (section 3.1.3.2) cherche à pallier ces différents inconvénients.

3.1.2.3 Le scanner moléculaire

3.1.2.3.1 Introduction

Un prolongement de ce travail a été réalisé par Markus Muëller durant sa thèse (Muller M. et al., 2002; Muller M., 2003; Muller et al., 2002). Le sujet portait sur le développement d'un outil d'analyse des données produites par la technique du scanner moléculaire (Bienvenut et al., 1999; Binz et al., 1999). Le principe de l'approche du scanner moléculaire est d'éviter la tâche difficile de la détection et sélection de spots après séparation des protéines sur un gel d'électrophorèse et coloration. Un autre travail avait été mené dans ce sens par Langen et al. (Langen H., 1998) dans lequel le gel était découpé systématiquement en petits rectangles et chacun d'eux soumis aux étapes de digestion et d'acquisition de spectre de masses. Dans l'approche du scanner moléculaire, la digestion se fait dans le gel lui-même puis l'ensemble des échantillons présents dans le gel et digérés est transféré, chaque échantillon conservant sa position dans le gel, sur une membrane PVDF pour être soumis à la mesure par un spectromètre de masse (MALDI-TOF). Le spectromètre effectue une série de mesures réparties régulièrement en forme de grille sur toute la surface de la membrane et produit une carte 2D de spectres. Un même spot est ainsi mesuré plusieurs fois et produit des données corrélées (puisque présente dans plusieurs spectres). Nous avons développé un outil dédié à ce type de données permettant une identification par PMF de toutes les protéines présentes sur le gel et tirant parti des propriétés spécifiques de la carte de spectres pour améliorer la qualité des identifications. L'outil d'analyse des données de scanner moléculaire comporte quatre parties : (1) la détection et l'extraction du bruit, (2) le clustering des masses de même distribution, (3) la calibration et (4) l'identification à partir des clusters de masses.

3.1.2.3.2 La détection et l'extraction du bruit

L'avantage majeur de la technique du scanner moléculaire est de pouvoir étudier la distribution des masses sur l'ensemble de la membrane. Nous avons conçu des outils de visualisation de cette distribution permettant de mettre à jour des différences fondamentales dans la répartition sur la membrane de certaines masses. La figure 14 (a) montre un exemple de telles répartitions sur la membrane[19] pour des masses comprises entre 800 et 1000 Da et la figure 14 (c) un exemple de l'occurrence de ces masses dans l'un des spectres du scanner. En tirant parti de la corrélation locale attendue pour les données peptidiques (un même spot est représenté dans plusieurs spectres contigus sur la membrane), un premier filtrage des masses les plus pertinentes est réalisé en sélectionnant les masses qui sont aussi présentes dans leur environnement immédiat (les

[19] Les deux dimensions de la base de ces graphes correspondent à la séparation des protéines en fonction de leur masse moléculaire et de leur point isoélectrique.

spectres contigus). Ce filtrage est visible sur la figure 14 (b). Nous avons également pu constater que les masses correspondant à des contaminents (comme la trypsine utilisée par la digestion ou la kératine humaine) ont des distributions homogènes sur toute la membrane (masses 804.419 Da et 820.413 Da de la figure 15 par exemple) alors que les masses peptidiques sont beaucoup plus localisées (masses 999.558 Da, 1254.742 Da, 1155.684 Da et 1193.623 Da de la figure 15 par exemple). Nous avons construit un algorithme utilisant cette propriété pour déterminer quelles sont les masses correspondant à des contaminents et donc quelles masses il est possible de retirer sans risque de la liste des masses pour l'identification. Il fonctionne par une division du plan de la membrane en petit rectangle et par l'analyse de la valeur moyenne de l'intensité de chaque masse dans chaque rectangle par rapport à sa valeur moyenne sur toute la membrane. Cet algorithme s'est avéré tout à fait à même de détecter tous les contaminents majeurs sur nos données de scanner moléculaire.

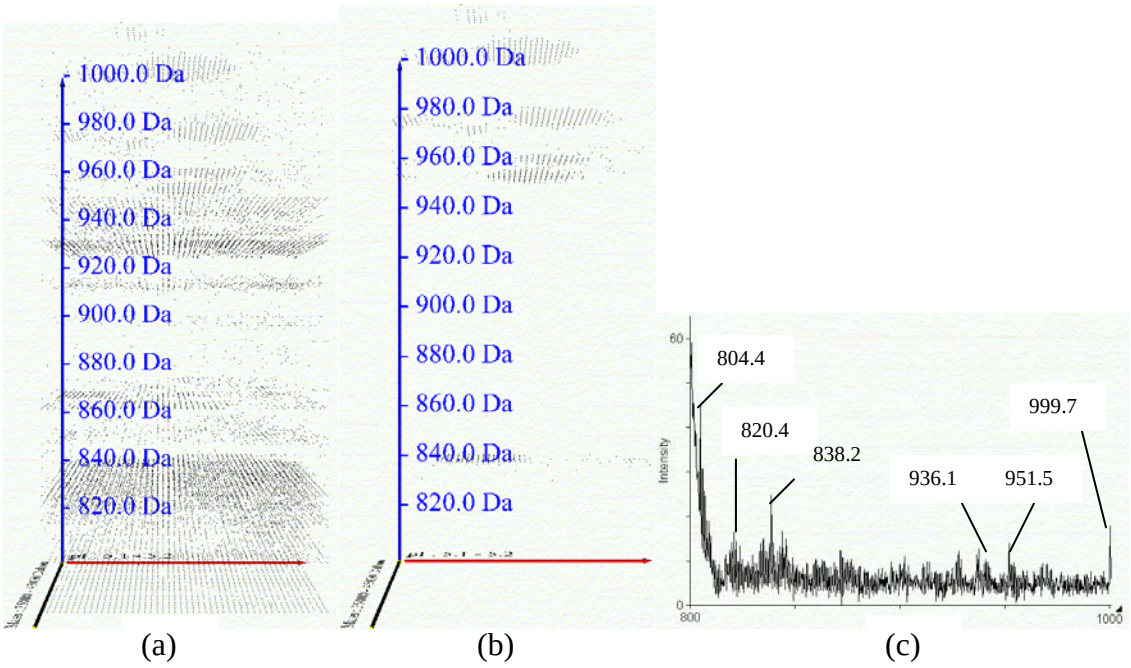


Figure 14. Distribution des masses comprises entre 800 et 1000 Da sur une membrane mesurée par le scanner moléculaire pour du plasma humain (a) sans filtrage, (b) après filtrage des masses sans voisin. Exemple de l'un des spectres mesurés pour les valeurs de masses considérées.

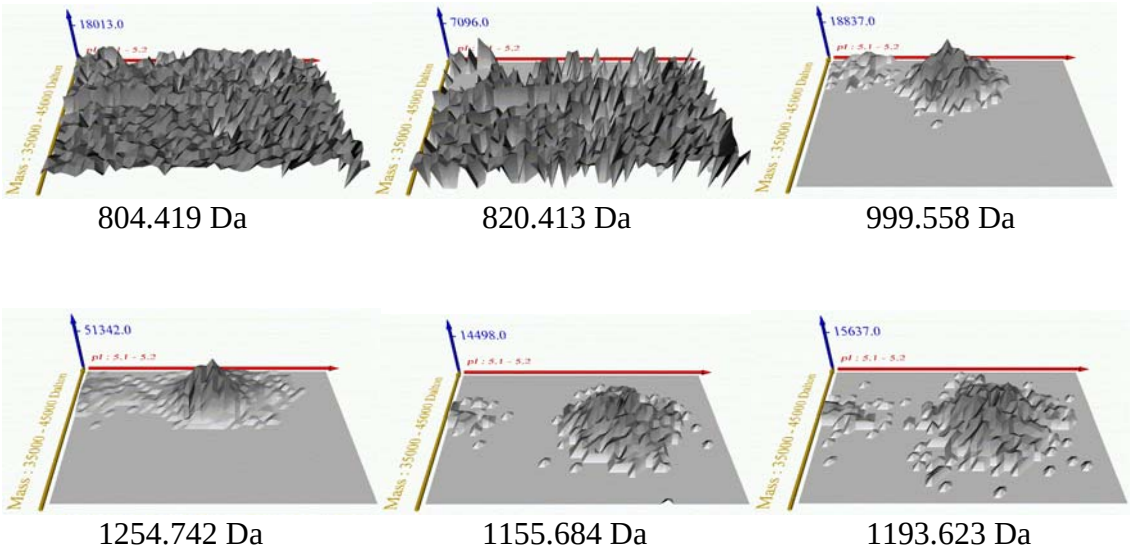


Figure 15. Exemples de la distribution sur la membrane de plusieurs masses. Les masses 804.419 et 820.413 correspondent à des contaminents, les autres sont potentiellement des masses peptidiques.

3.1.2.3.3 Clustering des masses de distributions similaires

Les masses peptidiques provenant de différentes protéines (et donc de différentes régions de la membrane) ont des distributions d'intensité également différentes. Par exemple dans la figure 15 les masses 999.558 Da et 1254.742 Da proviennent de la protéine METK_ECOLI et les masses 1155.684 Da et 1193.623 Da proviennent de la protéine ALLC_ECOLI. Nous avons développé un algorithme de clustering qui rassemble les masses ayant des distributions d'intensités similaires sur la membrane. Les clusters ainsi formés vont correspondre à l'ensemble des masses appartenant à une même protéine localisée dans un même spot[20]. Chaque distribution d'intensité sur la membrane I_i correspondant à la masse M_i est lissée avec une gaussienne bidimensionnelle dont les paramètres de déviations standards σ_x et σ_y sont donnés par l'utilisateur autour des valeurs typiques d'extension de spot en x (point isoélectrique) et y (masse moléculaire). Deux masses peptidiques appartenant à une même protéine vont alors avoir des distributions lissées de forme similaire et des maximums situés dans deux positions très rapprochées de la membrane. L'algorithme de clustering produit pour chaque cluster de masses C_i et chaque masse $j \in C_i$, la liste L_j des points de maximums d'intensité de ces masses. La figure 16 présente le résultat de l'algorithme de clustering appliqué à des données de scanner moléculaire pour du plasma humain. Les sommets correspondent au maximum des distributions lissées. Les spots des protéines AACT_HUMAN (1), A2GL_HUMAN (2), A1AT_HUMAN (3), VTDB_HUMAN (4) et HPT1_HUMAN_2 (7) sont clairement séparés alors que ceux de la protéine ALC1_HUMAN (5) sont distribués sous la forme d'un *train de spots* pouvant être dû à une série de variants de la

[20] Une protéine donnée peut être située dans plusieurs spots distincts si elle est présente sous différentes formes. Dans ce cas, les listes de masses pourront être également différentes.

protéine. La protéine FIBG_HUMAN (6) a, elle, une distribution de spots plus éclatée, peut-être due à des problèmes expérimentaux.

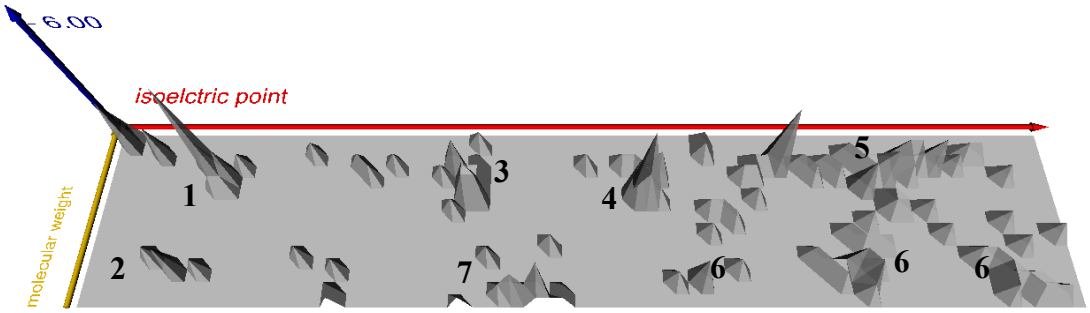


Figure 16. Visualisation des centres des clusters découverts pour une expérience de scanner moléculaire sur du plasma humain.

Un deuxième algorithme rassemble ensuite les masses correspondant à une même protéine et éventuellement présente dans plusieurs spots par un clustering hiérarchique cherchant à construire des distributions gaussiennes lissées à partir des distributions des masses des spots. Le résultat est une liste G de groupes de masses et de centres C_{Gi} de la distribution de toutes les masses contenues dans chaque groupe G_i. Ainsi, chaque groupe G_i doit contenir toutes et seulement toutes les masses appartenant à une même protéine éventuellement présente sous différentes formes.

3.1.2.3.4 La calibration

Le problème de calibration dans la technique du scanner moléculaire est plus complexe que dans le cas de mesure de spectres sur des échantillons indépendants. En effet, les mesures effectuées par le spectromètre sont très sensibles à de faibles variations de la structure de la membrane. De ce fait, il existe une grande variation de la calibration des spectres sur la membrane complète. Les masses de tous les groupes de masses doivent être recalibrées avant d'être utilisées pour l'identification.

Les masses détectées comme du bruit dans la première étape sont reprises pour détecter des masses pouvant correspondre à celles des peptides de la trypsine. Celles ainsi repérées sont placées dans une liste des *calibrants*. Puis tous les spectres de tous les groupes sont recalibrés. Le processus de calibration sur chaque spectre cherche à optimiser deux objectifs : rapprocher au maximum les masses du spectre similaires aux masses des calibrants de celles-ci et minimiser le nombre de masses du spectre dont les valeurs ne sont pas proches d'une valeur attendue. Ce dernier critère provient de l'observation des valeurs prises par les masses peptidiques, dans la base de données Swiss-Prot, qui sont centrées autour des valeurs discrètes (Gay S. et al., 1999). La méthode d'optimisation utilisée est basée sur la méthode de Powell (Press et al., 1995) adaptée pour prendre en compte le deuxième critère. C'est cette liste de masses recalibrées qui est utilisée pour l'identification.

3.1.2.3.5 L'identification à partir des groupes de masses

L'idée est ici de tirer parti de l'information de distribution des masses pour améliorer l'identification par PMF. Au cours de l'identification par PMF les masses d'une même protéine de la base de donnée qui auront été appariées à des masses expérimentales ayant des distributions dans la membrane similaire vont amplifier le score de cette protéine. Une nouvelle formule de score combiné S_c est définie pour toute position *pos* de la membrane par :

$$S_c(P_i, pos) = S_{PMF}(P_i, pos) \cdot S_{sim}(P_i, pos)$$

avec S_{PMF} le score d'identification PMF de SmartIdent qui produit également la liste $L(P_i,pos)$ des masses appariées, S_{sim} le nouveau score prenant en compte la similarité des distributions et P_i une protéine de la base de donnée. Le calcul de la valeur de $S_{sim}(P_i,pos)$ se fait par l'algorithme 1 :

$S_{sim}(P_i, pos) \leftarrow 0$
for each mass $m_j \in L(P_i,pos)$
$S_{sim}(P_i, pos) \leftarrow S_{sim}(P_i, pos) + \sum_{C_{Gi} \in G} \sum_{l \in L_j} \exp\left(-\frac{1}{2} d^2(l, C_{Gi}, \boldsymbol{\sigma})\right)$
end
$S_{sim}(P_i, pos) \leftarrow \frac{S_{sim}(P_i, pos)}{|L(P_i, pos)|}$

Algorithme 1. Calcul du score de similarité $S_{sim.}$

Un exemple de l'efficacité de ce nouveau score S_c est donné dans la figure 17 représentant le nombre de peptides appariés d'une protéine donnée pour chaque position de la membrane et pour les scores S_{PMF} et S_c. La localisation de la protéine est ainsi beaucoup mieux délimitée et les faux positifs éliminés.

Une dernière étape cherche à déterminer si un score donné est suffisant pour que la protéine soit considérée comme identifiée avec un bon niveau de confiance. Malheureusement, du fait de la forte dépendance du score aux nombres de masses impliquées, il est difficile de spécifier une valeur seuil pour le score qui soit valable pour toute la membrane. Pour éviter ce problème, un autre score, S_{tot}, a été défini qui donne le score d'une protéine pour une région complète de la membrane correspondant à la couverture d'un groupe de masses. On définit une région R_k correspondant au groupe de masses C_{Gk} par :

$$R_k = \left\{pos \mid I_i^{pos} > seuil, i \in C_{Gk}\right\}$$

avec I_i^{pos} la valeur de l'intensité de la masse *i* à la position *pos*. On peut alors déterminer la valeur de S_{tot} par :

$$S_{tot}(P_i, C_{Gk}) = \sum_{pos \in R_k} S_c(P_i, pos)$$

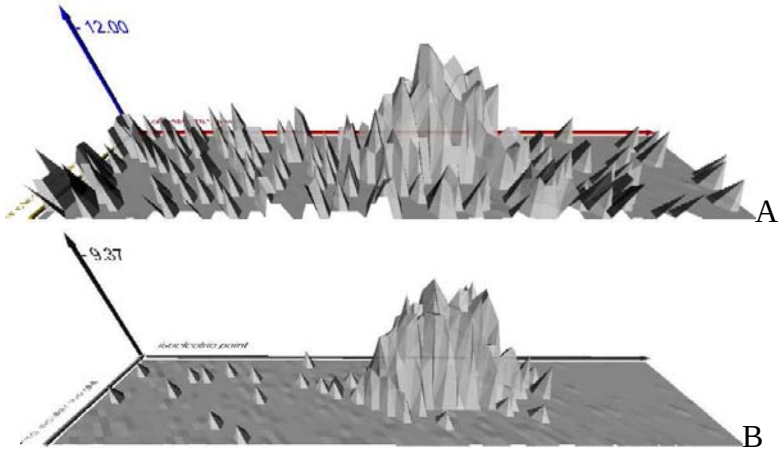


Figure 17. Nombre de peptides d'une protéine appariés pour chaque position de la membrane, A avec le score S_{PMF}, B avec le score S_c.

Pour chaque groupe de masses une liste de protéines potentiellement identifiées est produite. Il faut maintenant déterminer lesquelles dans cette liste sont effectivement des identifications fiables. Si l'on considère qu'au maximum trois protéines peuvent effectivement être présentes dans une région R_k de la membrane, on retire de la liste des protéines identifiées les trois protéines ayant le score le plus élevé. Puis on calcule la valeur moyenne $\overline{S_{tot}}$ et la déviation standard σ_{tot} du score des protéines restantes. On considère finalement comme identifiées de façon sûre les protéines de la liste complète telles que le Z-score de leur score S_{tot} est suffisamment haut :

$$\frac{\left| S_{tot} - \overline{S_{tot}} \right|}{\sigma_{tot}} > \varepsilon$$

Un exemple de la distribution comparée des scores S_{tot} obtenus à partir des scores S_c et S_{PMF} pour le spot contenant la protéine A2GL_HUMAN est donné dans la figure 18.

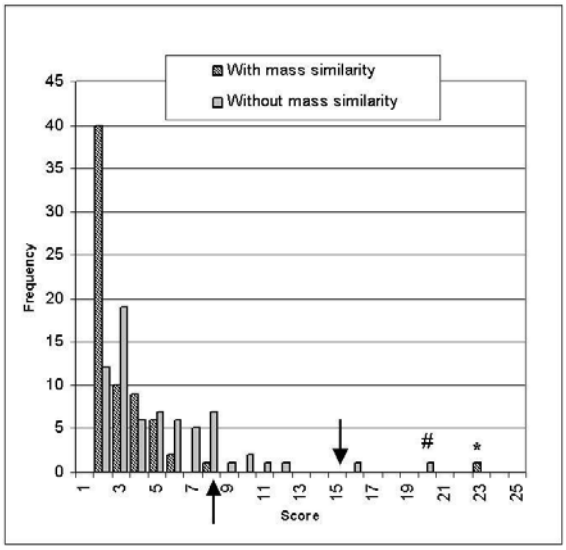


Figure 18. S_{tot} calculé pour les scores S_c et S_{PMF} pour toutes les protéines de la liste d'identification du spot contenant la protéine A2GL_HUMAN. Les valeurs obtenues par cette protéine sont marquées d'un # pour le score S_{PMF} et d'une * pour S_c. Les flèches indiquent les seuils d'identification à 90%, la plus basse pour le score S_c et la plus élevée pour le score S_{PMF}.

3.1.2.3.6 Résultats

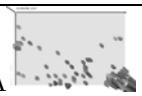
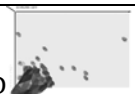
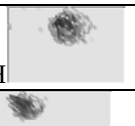
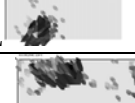
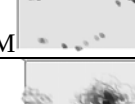
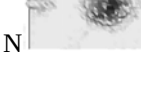
La qualité de la discrimination du score apparaît dans la valeur du Z-score obtenu par la bonne protéine. La table 3 présente la valeur du Z-score des bonnes protéines dans leur propre spot pour une expérience de scanner moléculaire sur du plasma humain montrant une nette amélioration du pouvoir de discrimination du nouveau score.

Protéine	Nouveau Z-score	Ancien Z-score
A2GL_HUMAN	33.05	6.49
HPT1_HUMAN_2	47.32	16.82
A1AT_HUMAN	76.96	19.05
FIBG_HUMAN	214.58	65.11
VTDB_HUMAN	52.88	16.3
ALC1_HUMAN	117.76	34.86

Table 3. Z-score des protéines du plasma humain avec le score S_c (nouveau) et le score S_{PMF} (ancien).

Deux expériences de scanner moléculaire ont été menées, une sur du plasma humain et l'autre sur un échantillon d'*E. coli.* Les résultats sont présentés dans la table 4 pour *E. coli* et la table 5 pour le plasma humain. Dans les deux cas, des protéines ont pu être identifiées par l'approche du scanner moléculaire alors qu'elles n'apparaissaient pas

après coloration des gels et donc ne pouvaient pas être identifiées par une approche classique.

Régions des groupes de masses	Score de discrimination (Z-score)	Masses expérimentales	Autre identification possible
 A		928.616 1060.315 1107.729 1323.094 1335.096 1443.048 1481.075	
 B	GSHB_ECOLI 32.23379	1388.031 1981.396 1997.287 2249.344 2250.249*	
 C		1262.957 1354.945 1433.059	
 D	POTF_ECOLI 90.90696	1060.315 1113.796 1278.042 1327.908 1354.945 1994.260 2051.662 2113.384	PDXB_ECOLI
 E		1498.316 1687.317	
 F	GALU_ECOLI 69.92648	1127.735 1157.763 1282.933 1489.102 1501.488 1527.101 1563.050 2788.512	
 G		1083.725 1534.080 1547.055 1598.082 1994.260 2020.501 2499.898 2500.516 2787.943	
 H	METK_ECOLI 370.18707	951.627 999.673 1108.664 1150.274*1211.706 1254.892 1255.427*1278.042 1292.850 1367.877 1466.093 1491.149 1552.897 2675.511*2737.503	
 I	6PGD_ECOLI 269.20128	1029.662 1056.639 1320.719 1364.904 1368.847 1985.105 2023.126 2403.325*2532.353 2935.980	6PG9_ECOLI
 J	ILVC_ECOLI 89.56738	1153.660 1225.856 1337.968 1339.897 1649.259 1687.317 1791.245 2232.836*2360.942*3150.692*	
 K		974.725 999.673 1111.724 1570.176	
 L	YJDA_ECOLI 36.57042	960.629* 998.600 1289.840 1399.028 1435.926 1994.260 2249.344	
 M	6PGD_ECOLI 58.44720	843.487 937.188 1060.315 1183.266 1496.073	ACEA_ ECOLI
 N	PGK_ECOLI 425.53572	928.616 954.447 969.636 975.651 991.548* 1007.598*1015.669*1021.640*1074.336*1135.717 1155.794 1161.834*1179.841 1182.309 1193.764 1243.818 1366.165 1427.924 1464.943 1493.997 1498.316 1546.299 1585.044 1875.130 1913.091 1992.255 1993.255 1998.214 2009.215 2025.211* 2031.145*2032.137*2040.135 2041.133 2060.118 2061.150 2126.290 2379.303 2380.238 2465.500 2466.502*2503.468 2504.503 3274.918 3275.725	

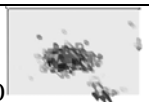
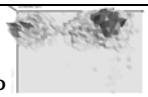
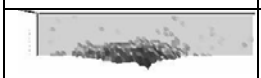
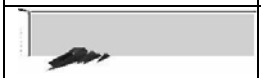
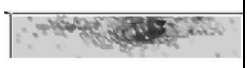
Régions des groupes de masses	Score de discrimination (Z-score)	Masses expérimentales	Autre identification possible
 O	ENO_ECOLI 56.68265	806.462 1018.539 1093.217 1189.786 1296.899* 1459.993 1531.065 1606.101 1644.062 1805.126* 1819.255 1929.169 1935.327 1967.115 1995.910 2075.356 2113.384	
 P	IDH_ECOLI 1450.03694	829.318 861.310 951.627 1006.600 1008.617 1024.608*1030.614 1040.629*1060.315*1063.325* 1124.744 1139.675 1171.892 1177.689 1206.759 1254.892*1373.839 1439.144 1500.899 1540.881 1586.087 1601.172 1615.096 1638.289 1663.043 1675.063 1689.099 1789.071 1805.126 1820.986 1869.158*2086.349 2202.232 2263.277 2304.342 2320.335 2343.299 2432.417 2494.363 2495.365*	
 Q	EFTU_ECOLI 76.54662	862.429 1073.628 1243.818*1586.087 1804.066 1962.181 1963.266 1965.085 2117.358 2466.502* 3276.762	PGK_ECOLI

Table 4. Identification pour l'expérience de scanner moléculaire d'*E. coli.* Les masses soulignées sont celles qui ont été appariées pour la protéine identifiée. Les masses marquées d'une * sont celles pour lesquelles il existe une modification post-traductionnelle prédite par l'outil FindMod (http://www.expasy.org) qui peut s'apparier. Toutes les protéines identifiées ont une valeur de discrimination supérieure à 30.

Régions des groupes de masses	Score de discrimination (Z-score)	Masses expérimentales	Autre identification possible
 A	HPT2_ HUMAN_2 49.07	862.354 920.527 978.567* 980.589 994.568* 996.546 1010.557 1012.566 1203.736 1495.889 1708.056 1835.181 1851.250	HPT1_ HUMAN_ 2
 B	ZA2G_HUMAN 71.06	974.568 990.540 1408.768 2403.346	
 C	A2GL_HUMAN 32.95	968.558 989.636 1006.514 1152.663 2959.091	
 D		1186.792 1224.787 1513.001 1808.232 2229.564	
 E	VTDB_HUMAN 107.83	934.575 950.556 1032.632*1213.792*1254.897 1266.914 1268.915*1326.945 1340.945*1354.836* 1388.918 1404.913 1409.845 1437.941 1513.001 1530.081 1695.169 1733.212 1734.116 1951.302 2157.306 2328.559	ZN93_ HUMAN
 F	A1AT_HUMAN 78.16	833.374 852.512 1015.706 1053.689 1076.597 1247.738 1263.698 1285.720*1403.819 1642.094 1842.151*1856.153 1857.132*1880.238*2090.352 2162.457 2757.752 2758.840	
 G		1032.632 1641.097 1858.096	

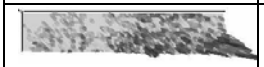
 H		924.457 927.545 1080.695 1081.235 1094.696 1102.662 1118.613 2035.222 2048.097 2049.167 2081.077 2093.053 2094.058 2096.115 2097.037 2109.124 2110.111 2111.093	
 I	PSE2_HUMAN 50.33	833.374 871.323 877.224 1050.253 1066.239 1837.056*	
 J	FIBG_HUMAN 218.93	923.593*1036.647 1152.663*1161.744*1293.935 1513.962 1546.045 1551.964 1683.261 1697.305 1894.260 1899.293 1900.321 1932.267 1995.267 1996.337 2012.268 2026.268 2034.300 2050.234* 2051.267 2286.671*2417.554 2520.730 2521.763	
 K	ALC1_HUMAN 115.30	830.468 844.507 868.425 894.536 896.556 900.490 910.548* 927.545 931.585 1034.642* 1048.624*1074.593 1117.723 1150.610 1153.695 1223.752 1225.802 1239.789 1243.776* 1257.802*1269.814 1338.841 1352.866 1353.863 1375.802 1540.918 1546.965 1578.899*1625.088 1819.173 1836.216 1850.211*1864.275*1882.268 1896.248 1899.293 1952.234 2337.387 2338.410* 3168.716 3169.690*	ALC2_ HUMAN
 L	ALBU_HUMAN 85.22	915.465 960.658 1138.605 1443.822 1468.060 1481.923* 1512.045 1549.958* 1560.058 1624.084 1640.160 1654.209* 1668.235* 1900.321* 1933.226* 2051.267* 2064.220* 2599.696	

Table 5. Identification pour l'expérience de scanner moléculaire de plasma humain. Les masses soulignées sont celles qui ont été appariées pour la protéine identifiée. Les masses marquées d'une * sont celles pour lesquelles il existe une modification post-traductionnelle prédite par l'outil FindMod (http://www.expasy.org) qui peut s'apparier. Toutes les protéines identifiées ont une valeur de discrimination supérieure à 30.

3.1.3 Identification par spectrométrie de masse en tandem

3.1.3.1 Etat de l'art

3.1.3.1.1 Introduction

Les spectres MS/MS sont également obtenus par spectrométrie de masse d'un échantillon de protéines. Avant l'analyse, les protéines sont digérées avec une enzyme spécifique ; puis les peptides résultants sont introduits dans le spectromètre de masse. L'obtention des spectres MS/MS se fait alors en deux étapes : premièrement, les peptides sont ionisés et passent dans un premier analyseur. Deuxièmement, à tour de rôle, chaque peptide est isolé, fragmenté, et envoyé dans un deuxième analyseur qui mesure le rapport de la masse sur la charge (*m/z*) des fragments obtenus et produit un spectre MS/MS. Ce dernier est composé de la masse du peptide source, appelée masse parente, et d'un signal continu qui est traité et transformé en une liste de pics. Pour identifier les protéines sources (c'est-à-dire les protéines de l'échantillon), il faut alors parvenir à associer chaque spectre MS/MS avec une séquence peptidique connue. Dans le cas le plus simple, l'échantillon ne contient qu'une espèce de protéine, et cette dernière est représentée dans la base de données de séquences utilisée pour l'identification. Les choses se compliquent lorsque la protéine source n'est pas présente dans la base de données. Trois stratégies peuvent alors être adoptées : utiliser des séquences obtenues à partir de données

génomiques, chercher à caractériser la ou les protéines d'origine à l'aide de protéines homologues ou extraire une courte séquence d'acides aminés à partir du spectre et construire une sonde pour cloner le gène correspondant.

Le choix des programmes d'identification à partir de données MS/MS est vaste. Les premières approches ont été décrites dans les années 80 et depuis de nombreux autres algorithmes ont été décrits, plus performants et plus spécifiques. Ils sont souvent conçus pour un problème précis, selon par exemple le type de base de données (protéique ou génomique), ou la quantité de spectres à analyser (méthodes autonomes ou nécessitant l'intervention d'experts), ou encore la présence éventuelle de modifications ou de mutations sur le peptide source.

Il y a plusieurs manières d'analyser un spectre MS/MS. Une première consiste à inférer de l'information sur la séquence du peptide source sans l'aide d'une base de données. On parle alors de *de novo* sequencing. La séquence obtenue peut ensuite être utilisée de plusieurs manières : on peut l'aligner avec des séquences théoriques afin de corréler le spectre avec une protéine connue (la protéine source ou le cas échéant, une protéine homologue), ou l'on peut construire une sonde dans le but de cloner le gène correspondant. Une deuxième manière d'aborder un spectre MS/MS est de le comparer directement avec des spectres théoriques calculés à partir de séquences provenant de banques de données. Cette technique a récemment été dénommée peptide fragment fingerprinting (PFF), par analogie au peptide mass fingerprint. Les différentes méthodes décrites dans la littérature rentrent soit dans la première catégorie (*de novo* sequencing), soit dans la deuxième (PFF) ou combinent les deux techniques.

3.1.3.1.2 Le *de novo* sequencing

Historiquement, l'intérêt a d'abord porté sur le *de novo* sequencing, qui représentait une alternative attrayante au séquençage de protéines par dégradation d'Edman, avec deux avantages principaux : la méthode était plus rapide et ne nécessitait pas un degré de purification élevé des protéines. Les premiers programmes de *de novo* sequencing datent des années 80. Leurs algorithmes consistent à générer des peptides théoriques de masse similaire à la masse parente observée en combinant les acides aminés; puis ils calculent les masses théoriques des fragments de ces peptides, et les comparent avec les masses expérimentales du spectre à analyser (Hamm C.W. et al., 1986; Matsuo T. et al., 1981; Sakurai T. et al., 1984). Certaines stratégies, comme l'utilisation d'information sur la composition en acides aminés du peptide source, sont proposées afin de réduire le nombre de combinaisons possibles, qui augmente exponentiellement avec la masse parente considérée. Cette approche, assez simpliste, a rapidement évolué. Dès la moitié des années 80, les programmes (Ishikawa K. and Niwa Y., 1986; Siegel M.M. and Bauman N., 1988; Yates J.R. et al., 1991) utilisent une approche incrémentale : les séquences sont construites en ajoutant itérativement des acides aminés à un ensemble de sous-séquences; seules les extensions validées par la présence de pics dans le spectre sont considérées. Mais les méthodes incrémentales sont particulièrement sensibles à la qualité des spectres. En effet, il suffit d'un gap de plus de 2 acides aminés dans le spectre, en d'autres termes de deux positions successives non

fragmentées sur le peptide source, pour que la séquence correcte soit définitivement mise de côté. En 1990, Bartels (Bartel C., 1990) propose de représenter le spectre sous la forme d'un graphe appelé *graphe de spectre* (*spectrum graph*). Désormais, faire du *de novo* sequencing revient à parcourir un graphe dont les nœuds, formés à partir des *m/z* du spectre, représentent des masses potentielles de fragments du peptide source. Lorsque deux nœuds diffèrent par la masse d'un ou plusieurs acides aminés, ils sont reliés par un pont étiqueté par le ou les acides aminés correspondants. Cette structure est dès lors très largement adoptée (Dancik V. et al., 1999; Fernandez-de-Cossio J. et al., 1995; Hines W.M. et al., 1991; Taylor J.A. and Johnson R.S., 1997). En 2001, Chen et al. (Chen T. et al., 2001) proposent d'utiliser la programmation dynamique pour parcourir plus efficacement le graphe.

Après avoir inféré de l'information sur la séquence du peptide source par *de novo* sequencing, il faut l'utiliser pour extraire d'une base de données le ou les peptides candidats à l'identification. En 1997, Taylor et al. s'inspirent de l'algorithme d'alignement de séquences Fasta et implémentent CIDentify (Taylor J.A. and Johnson R.S., 1997). Ce dernier a la particularité de pouvoir prendre en entrée un ensemble composé de plusieurs séquences; chaque séquence est alignée contre les séquences de la base de données, et le score obtenu pour chaque séquence est ajouté au score global de la séquence théorique courante. CIDentify prend en compte certaines caractéristiques propres à l'alignement de séquences obtenues par *de novo* sequencing avec des séquences théoriques. Ainsi, il autorise l'alignement des acides aminés différents mais dont les masses concordent, comme la leucine et l'isoleucine; il autorise également l'alignement d'un acide aminé avec deux autres, à nouveau pour autant que les masses totales concordent. Enfin, il prend en compte les masses correspondant à des combinaisons de plusieurs acides aminés. Depuis CIDentify, d'autres programmes d'alignement spécifiques pour les données séquences obtenues par *de novo* sequencing à partir de spectres MS/MS ont été décrits : MS-BLAST (Shevchenko A. et al., 2001), MS-Shotgun (Huang L. et al., 2001), FASTS (Mackey A.J. et al., 2002), MultiTag (Sunyaev S. et al., 2003) et OpenSea (Searle B.C. et al., 2004). Ces programmes sont principalement utilisés pour corréler le spectre expérimental à des protéines homologues. L'intérêt de ces algorithmes est qu'ils permettent d'identifier des peptides portant des modifications ou mutations non attendues puisqu'ils autorisent un ou plusieurs mésappariements lors de l'alignement et ne filtrent pas les séquences candidates en fonction de leur masse parente.

3.1.3.1.3 Le peptide fragment fingerprint

Le principe de base du PFF est de créer des spectres MS/MS virtuels à partir de séquences protéiques ou génomiques répertoriées dans les bases de données et de les comparer avec le spectre expérimental. Le degré de similitude entre le spectre expérimental et chaque spectre théorique est mesuré par un score, plus ou moins évolué selon les méthodes. Le peptide obtenant le meilleur score est considéré comme étant celui qui représente le spectre expérimental, pour autant que le score obtenu dépasse un seuil de confiance.

La première méthode d'identification par PFF a été décrite en 1994 par Eng et al. (Eng J.K. et al., 1994). Leur algorithme, Sequest, utilise comme mesure de similarité une fonction de cross-correlation par produit des transformées de Fourier des listes de masses expérimentales et théoriques. Au cours des années suivantes, de nouveaux programmes sont décrits, dont PepFrag (Fenyo D. et al., 1998), Mascot (Perkins D.N. et al., 1999), SCOPE (Bafna V. and Edwards N., 2001), Sonar (Field H.I. et al., 2002), et PepProb (Sadygov R.G. and Yates J.R.III, 2003). Certains, comme Mascot et PepFrag, autorisent plusieurs types de recherche (MS, PFF et/ou recherche en utilisant une séquence d'acide aminé extraite *de novo* du spectre). Les scores employés varient également entre les programmes. Par exemple, Sonar utilise comme fonction de comparaison le produit vectoriel entre le spectre expérimental et chaque spectre théorique. Les spectres sont d'autant plus similaires que leur produit vectoriel est proche de 0. L'algorithme de Mascot a été modifié récemment pour permettre une identification en deux étapes (Creasy D.M. and Cottrell J.S., 2002). Les protéines identifiées lors du premier passage peuvent être réévaluées contre les spectres expérimentaux en autorisant une liste plus étendue de modifications chimiques et post-traductionnelles, ainsi que des mutations. Le principe d'identification en deux étapes est décrit notamment dans l'article récent de Craig et Beavis (Craig R. and Beavis R.C., 2003). On le retrouve dans d'autres méthodes, comme TANDEM, des mêmes auteurs (Craig R. and Beavis R.C., 2004) et Olav (Colinge J. et al., 2003), qui a la particularité de proposer une fonction de score de probabilité bayésienne incluant un grand nombre de paramètres différents. Il va de soi que l'efficacité de ces différentes méthodes dépend avant tout de la qualité de prédiction des spectres théoriques. De nombreuses études ont été accomplies dans cette voie. Certaines méthodes sont basées sur des observations statistiques (Breci L.A. et al., 2003) afin de dériver des règles à prendre en compte lors des prédictions de spectres. Elias et al. (Elias J.E. et al., 2004) tentent de modéliser les intensités attendues à l'aide d'un arbre probabiliste de décision.

De nos jours, l'accent est souvent mis sur la capacité des algorithmes à identifier des spectres MS/MS provenant de peptides modifiés ou mutés. En effet, la présence de modifications (chimiques ou post-traductionnelles) ou de mutations modifie les masses mesurées des fragments qui les portent et les spectres MS/MS résultants ont des groupes de pics qui sont décalés sur l'axe des *m/z* dans un sens ou dans l'autre selon la différence de masse provoquée par la modification ou la mutation. Toutes les méthodes de PFF décrites plus haut nécessitent de prévoir les modifications ou mutations à prendre en compte en les incluant dans la base de données, ce qui se solde inévitablement par une énorme augmentation du nombre de séquences théoriques à comparer. Certains algorithmes ont été spécifiquement conçus pour gérer ce type d'identification. Mann et Wilm en 1994 font figure de pionniers en définissant la notion de tag, comme une courte séquence flanquée d'une masse préfixe et d'une masse suffixe. Leur méthode d'identification consiste à inférer un tag à partir d'une région de haute qualité du spectre MS/MS, et de l'utiliser pour extraire d'une base de données les peptides qui matchent la séquence et au moins l'une des deux masses. Ce faisant, ils autorisent une ou des modifications dans les régions préfixes ou suffixes. Une fois extraits, les peptides candidats sont re-scorés contre le spectre expérimental, cette fois-ci en utilisant une méthode de PFF. Cette approche a été reprise récemment par Tabb (Tabb D.L. et al.,

2003) avec un score amélioré. Une méthode différente, nommée PEDENTA, a été décrite par Pevzner et al. (Pevzner P.A. et al., 2000) . Elle consiste à aligner, par programmation dynamique, les masses expérimentales et théoriques en autorisant un certain nombre de sauts, qui correspondent à d'éventuelles modifications ou mutations. Notre algorithme, Popitam (Hernandez P. et al., 2003), se propose de construire une liste de tags spécifiques pour chaque séquence de la base de données (en fait, ce sont les séquences qui guident l'extraction des tags), puis d'arranger les tags suivant des règles de compatibilités avant de leur attribuer un score. Il est présenté plus en détail dans la section suivante.

3.1.3.2 Popitam

La thèse de Patricia Hernandez porte sur la conception d'un outil d'identification, appelé Popitam (Hernandez P. et al., 2003), à partir de données de spectrométrie de masse en tandem. Basé sur l'extraction de tag, il utilise la différence de masse entre fragments ioniques pour extraire des tags potentiels des spectres MS/MS. Une telle approche apporte deux avantages majeurs. Premièrement, elle permet de contourner les problèmes de calibration des spectres MS/MS[21] car ce ne sont pas directement les masses des fragments théoriques qui sont comparées aux masses expérimentales mais des masses d'acides aminés qui sont comparées à des différences de masses expérimentales. Deuxièmement, elle permet d'identifier des peptides porteurs de mutations ou de modifications inconnues. Ce second avantage est tout particulièrement important puisque identifier des protéines porteuses de mutations ou de modifications inconnues est nécessaire pour découvrir la cause de nombreuses maladies. Il est tout particulièrement difficile de traiter ce problème de modifications par les méthodes de PFF car une modification sur un acide aminé va changer la masse observée (par rapport à la masse théorique attendue) de toute une série de fragments (ceux qui possèdent la modification) et non des autres. Bien entendu, rien ne permet de déterminer, en observant uniquement les pics du spectre, lesquels correspondent à un fragment qui porte une modification. Notre outil est dédié spécifiquement à ce problème difficile.

Chaque spectre à identifier est dans un premier temps transformé en graphe de spectre. Puis chaque séquence de la base de données est comparée avec le graphe. Le processus suivi par Popitam pour donner un score de similarité entre une séquence de la base de données et un spectre est le suivant (voir aussi la figure 19) : (1) extraire tous les tags du graphe compatibles avec la séquence en parcourant simultanément la séquence indexée et le graphe, (2) éliminer tous les tags redondants ou inconsistants avec les données expérimentales, (3) construire le graphe des compatibilités de tags, (4) chercher toutes les cliques de ce graphe, (5) donner un score à chaque clique, (6) le score de la séquence est le score obtenu par la meilleure clique. L'intérêt de l'utilisation des tags réside dans le fait que les régions situées entre les tags correspondent soit à un manque d'information dans le spectre, soit à des modifications ou des mutations sur le peptide source. Ainsi, en appariant des tags indépendants avec une séquence, on peut corréler un

[21] Problèmes de calibrations qui sont plus complexes à traiter que ceux du PMF du fait de la présence de fragments multiplement chargés (donc avec des valeurs de masse/charge qui ne sont pas à la même échelle) et du fait du mélange de fragments C-terminaux, N-terminaux et internes au peptide.

maximum d'informations expérimentales et théoriques en évitant les problèmes de décalage de masses liés aux modifications, ce que ne peuvent pas faire les approches PFF.

3.1.3.2.1 Construction du graphe de spectre

La première étape de Popitam consiste en la transformation des données du spectre MS/MS à identifier en une représentation structurée : le graphe de spectre (Fernandez-de-Cossio J. et al., 1995). Une complication des spectres MS/MS par rapport à ceux utilisés pour le PMF vient du fait que chaque pic peut être interprété comme plusieurs masses peptidiques possibles. En effet le principe du MS/MS est que le peptide mesuré est fragmenté par bombardement après avoir été chargé pour voler dans le spectromètre. Les points de coupures correspondent aux liaisons entre acides aminées plus ou moins quelques atomes, ce qui conduit à la production, pour une même liaison, à plusieurs types de fragments correspondant à plusieurs types d'ions. De plus les fragments peuvent être C ou N-terminaux et peuvent être porteurs d'un nombre variable de charges. Donc chaque pic du spectre doit être interprété de différentes façons pour déterminer toutes les masses de fragments peptidiques initiales qui ont pu produire ce signal. A chaque pic p_i d'un spectre S on fera donc correspondre une liste $l(p_i)$ contenant toutes les masses de fragments peptidiques pouvant en être à l'origine. L'ensemble des masses de fragments peptidiques possibles à partir d'un spectre S contenant n pics est noté $M_S = \{l(p_1), \ldots, l(p_n)\}$. C'est à partir de cette liste de masse M_S que le graphe de spectre G_S est construit.

A chaque masse de M_S correspond un et un seul nœud de G_S. Puis tous les nœuds dont la masse diffère par la masse d'un acide aminé (plus ou moins une erreur ε) sont reliés par un arc étiqueté par le nom de cet acide aminé. Il est également possible d'ajouter des arcs-doubles qui relient des nœuds dont la masse diffère de la masse d'une combinaison de deux acides aminés. L'ensemble des chemins possibles dans le graphe ainsi formé représente l'ensemble de successions d'acides aminés (appelé *tag*) pouvant être construites à partir du spectre. L'avantage des arcs-doubles est que si aucun des fragments correspondant à un acide aminé de la séquence peptidique complète n'est présent dans le spectre on peut construire un chemin passant par un arc-double qui permet de faire un "pont" au-dessus de l'acide aminé manquant et retrouver la séquence initiale. On pourrait étendre ce concept à des arcs-triples et plus mais le nombre d'arcs augmente exponentiellement avec la longueur des arcs et il n'est pas réaliste d'aller au-delà des arcs-doubles. La figure 20 montre un exemple de graphe construit à partir d'un spectre MS/MS du peptide LVNELTEFAK. Dans l'image de gauche on peut voir le chemin correspondant à la séquence peptidique complète et dans l'image de droite un autre chemin, passant par un arc-double et correspondant à une autre séquence peptidique. On peut donc constater que si le graphe de spectre représente bien la séquence et toutes les sous-séquences d'acides aminés que l'on peut constituer à partir du peptide d'origine, il représente aussi un très grand nombre d'autres séquences (mauvaises interprétations du spectre) n'ayant rien à voir avec la séquence d'origine. C'est ce qui rend difficile l'identification (et le *de novo* sequencing) à partir de données MS/ MS.

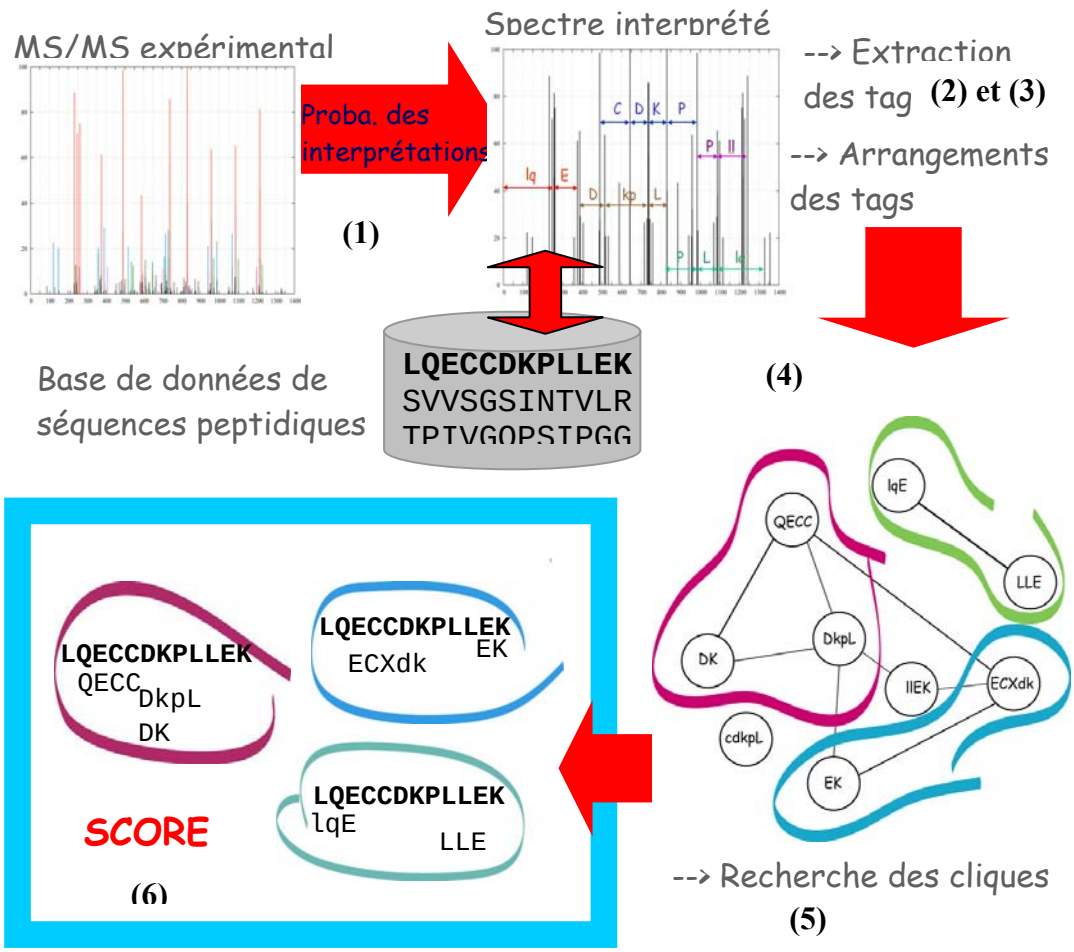


Figure 19. Processus général de Popitam pour l'identification d'un spectre MS/MS.

Une manière de limiter le nombre de tags pouvant être construits à partir du graphe est de limiter la taille du graphe. Cela peut se faire en jouant sur le nombre de nœuds et donc sur le nombre d'interprétations différentes à faire pour chaque pic du spectre. Nous avons choisi une approche par apprentissage ayant pour but de filtrer les interprétations des pics en fonction de la probabilité d'occurrences des différents types de fragments pour un spectromètre donné. En utilisant comme ensemble d'apprentissage une série de spectres produits par un spectromètre donné et clairement identifiés, on dispose d'une base annotée de fréquences d'occurrences des différents types de fragments. Nous construisons à partir de cette base une table de probabilité d'occurrence des différents types de fragments en fonction de l'intensité du signal dans le spectre et de la charge du fragment. Nous utilisons cette probabilité pour choisir les interprétations que nous utilisons pour construire les nœuds du graphe. Cette technique avait déjà été proposée dans (Bartel C., 1990). Nous l'étendons de façon à garantir la présence de types de fragments fréquents pour chaque classe d'intensité. En effet, des fragments, ayant même

une intensité de signal faible mais dont la présence est probable dans le spectre, sont une source d'information importante qu'il ne faut pas éliminer.

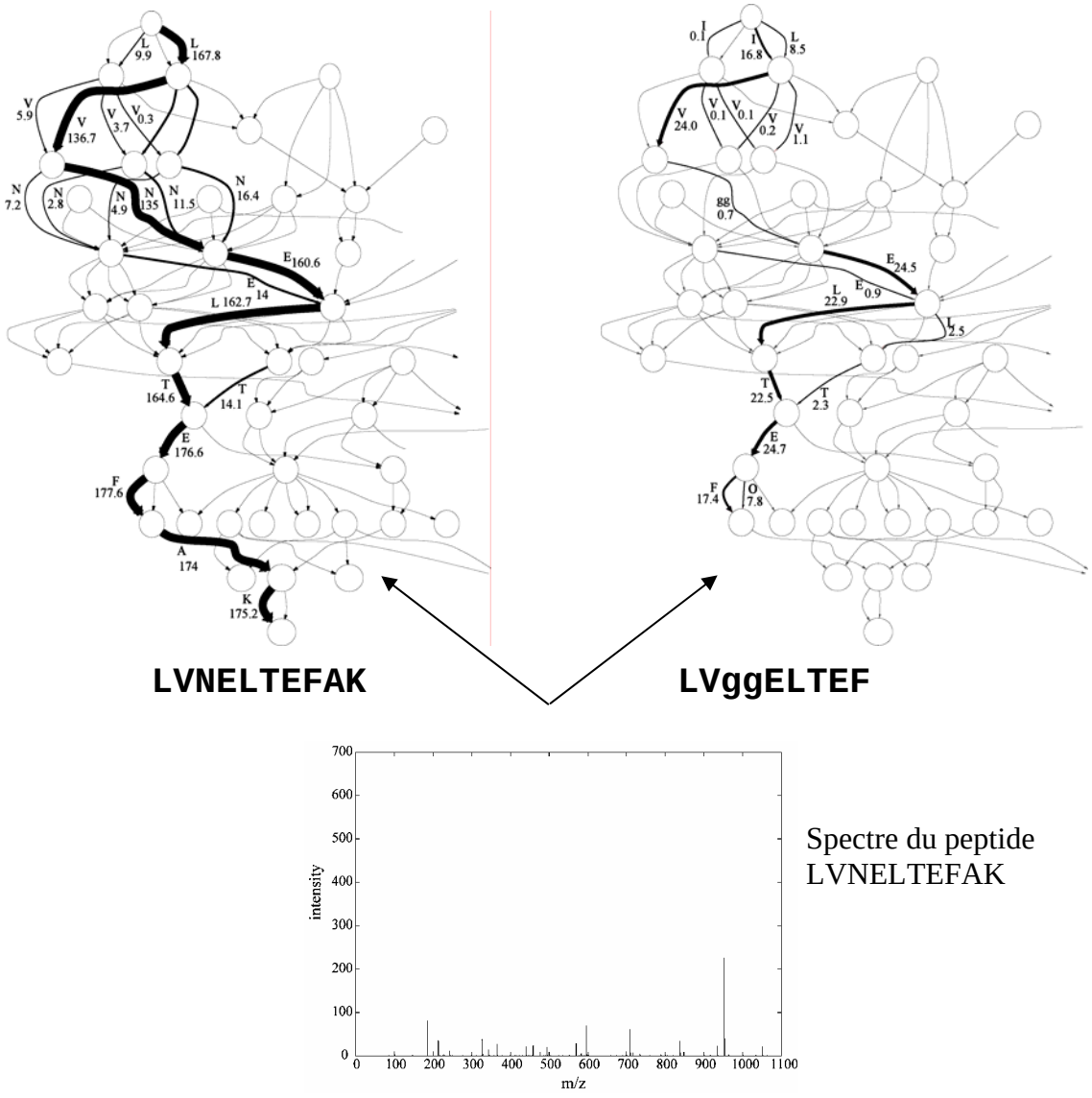


Figure 20. Exemple de graphe de spectre construit à partir d'un spectre MS/MS du peptide LVNELTEFAK. L'image de gauche montre le chemin correspondant à la bonne séquence et l'image de droite un chemin correspondant à une autre séquence du fait d'une mauvaise interprétation du spectre.

3.1.3.2.2 Extraction et élimination des tags

A la différence de toutes les méthodes de *de novo* sequencing basées sur l'utilisation du graphe de spectre, nous ne recherchons pas tous les tags ou toutes les séquences complètes contenues dans le graphe car leur nombre est extrêmement élevé. Par exemple, pour un spectre de 145 pics (valeur classique), si on ne considère, après filtrage des interprétations de pics, qu'un graphe contenant 68 arcs et 831 arcs-doubles (ce qui correspond à un filtrage assez drastique), on obtient de l'ordre de 2.10^{10} tags. Notre approche consiste à ne considérer que les tags qui sont compatibles avec une séquence théorique. Pour ce faire, pour toute séquence de la base de données, on extrait du graphe tous les tags inclus strictement dans la séquence. De façon à optimiser ce processus, chaque séquence de la base de données est indexée sous la forme d'un arbre des suffixes (Gras R., 1997; Ukkonen E., 1993; Weiner P., 1973) avant d'être comparée au graphe. Nous réalisons ensuite un parcours simultané de l'arbre et du graphe, extrayant du graphe tous les tags indexés dans l'arbre. Nous nous limitons également à des tags d'une longueur supérieure à une longueur seuil dépendant de la masse du peptide parent (de l'ordre de 3 ou 4 acides aminés). Un tag dans Popitam comporte plusieurs informations : une masse de début et une masse de fin toutes deux déterminées à partir des masses des pics contenues dans le premier et le dernier nœud du chemin du graphe constituant le tag, sa séquence d'acides aminés, sa positon dans la séquence théorique et la liste des nœuds du graphe inclus dans le chemin.

L'élimination des tags se fait selon plusieurs critères :

- Tous les tags (du point de vue de la liste des nœuds) entièrement inclus dans d'autres tags sont éliminés car redondants.
- On élimine les tags construits à partir de nœuds dont les masses proviennent d'un même pic expérimental. En effet, comme on l'a vu précédemment, un pic peut être interprété en plusieurs masses qui vont constituer plusieurs nœuds. Si un chemin du graphe conduisant à l'un des tags passe par au moins deux de ces nœuds, on a une incohérence et donc on élimine le tag.
- On élimine les tags se trouvant en début de séquence par appariements des caractères mais dont la masse de début n'est pas compatible avec un début de séquence.
- On élimine les tags se trouvant en fin de séquence par appariements des caractères mais dont la masse de fin n'est pas compatible avec une fin de séquence.
- On regroupe les tags qui utilisent chacun un nœud provenant d'une interprétation différente de deux pics correspondant au même fragment. En effet, un même fragment peptidique peut produire plusieurs pics. Ces pics, une fois interprétés, vont donner des nœuds de masses presque identiques (aux erreurs de mesures près) et vont donc pouvoir constituer des tags de même séquences mais utilisant des nœuds différents. Ces tags représentent en fait la même information mais peuvent également être vus

comme une confirmation d'une bonne interprétation de l'information spectrale. On appelle ces groupes de tags des *familles de tags*.

Une fois tous les tags possibles éliminés, on procède à la construction du graphe des compatibilités.

3.1.3.2.3 Graphe de compatibilité et cliques

Nous utilisons également une série de règles pour déterminer si deux tags sont compatibles. Ces règles sont basées sur la valeur respective des masses de début et de fin des deux tags et leur placement relatif sur la séquence peptidique par appariement de caractères. La figure 21 montre la liste des règles utilisées sur un exemple donnant toutes les situations rencontrées lors de l'appariement de deux tags sur la séquence peptidique (ici la séquence LQECCDKPLLEK). Une règle supplémentaire est utilisée : deux tags compatibles ne peuvent pas contenir de nœuds correspondant à un même pic expérimental. En utilisant ces règles, on constitue le graphe de compatibilité entre tags. Chaque tag est un nœud du graphe. On relie deux nœuds si aucune des règles ne rend les deux tags correspondants incompatibles. On recherche ensuite dans ce graphe toutes les cliques, c'est-à-dire tous les sous-graphes entièrement connectés. On commence la recherche par les cliques de taille 2 et puis on incrémente la taille des cliques jusqu'à ce qu'on arrive au nombre total de tags. Si, à une étape de cette recherche, on ne découvre pas de clique, on arrête la recherche, puisqu'on est garanti d'avoir trouvé la clique la plus grande. Bien que ce processus soit NP-complet, le filtrage des tags diminue fortement le nombre de tags considérés et les règles de compatibilité rendent improbables les cliques de grande taille. Cette approche reste donc applicable à des données réelles en un temps raisonnable.

3.1.3.2.4 Score du peptide

Un score est attribué à chaque clique mesurant la similarité entre la séquence de la base de données et les informations expérimentales liées aux tags composant la clique. Le score final de la séquence est le score maximum obtenu pour toutes les cliques. Pour déterminer la fonction de score utilisée nous avons suivi une approche ressemblant à celle utilisée pour SmartIdent (voir la section 3.1.2.2). L'objectif est de prendre en compte un maximum d'informations expérimentales et théoriques pour déterminer la mesure de similarité. Nous avons défini un ensemble de sous-fonctions de scores simples, chacune d'elles étant basée sur une information particulière. Puis nous avons construit notre fonction de score globale en combinant ces différentes sous-fonctions. Les sous-fonctions que nous avons utilisées jusqu'à présent sont :

- *CS1* : couverture de la séquence théorique par des acides aminés provenant d'arcs simples.
- *CS2* : couverture de la séquence théorique par des acides aminés provenant d'arcs-doubles.

- *CS3* : couverture globale de la séquence théorique.
- *PS1* : moyenne arithmétique des probabilités des interprétations des pics des nœuds constituant les tags de la clique.
- *PS2* : moyenne arithmétique des probabilités au carré des interprétations des pics des nœuds constituant les tags de la clique.
- *FS1* : moyenne arithmétique de la taille des familles de tags, des tags de la clique.
- *FS2* : moyenne arithmétique de la taille au carrée des familles de tags, des tags de la clique.
- *LS1* : moyenne arithmétique de la longueur des tags de la clique.
- *SS1* : nombre de tags dans la clique.

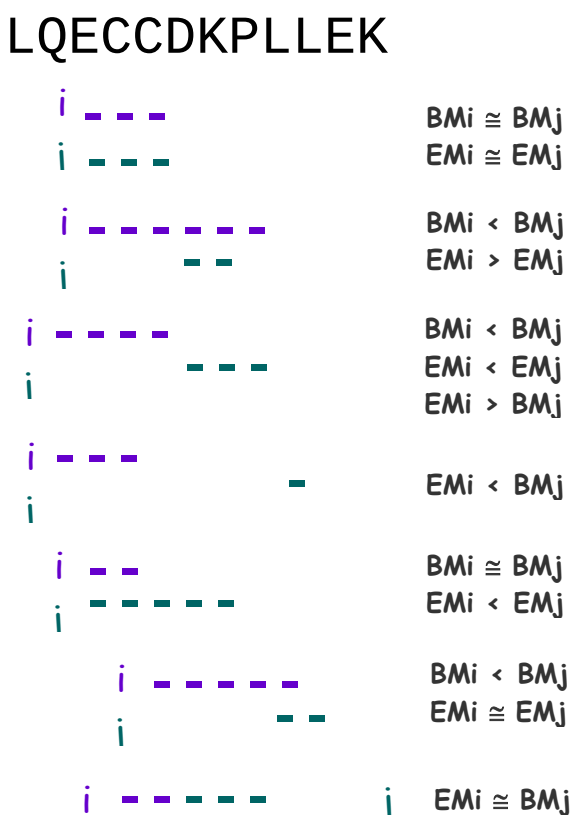


Figure 21. Règles de compatibilité entre deux tags *i* et *j* appariés à une séquence peptidique (ici LQECCDKPLLEK). Avec BMi la masse de début du tag *i*, EMi la masse de fin du tag *i*, BMj la masse de début du tag *j* et EMj la masse de fin du tag *j*.

Nous avons défini une première fonction de score empirique *SE* à partir de ces sous-fonctions :

$$SE = \frac{CS1 + CS2 + CS3}{3} \cdot \frac{PS1 + PS2}{2} \cdot \frac{FS1 + FS2}{2} \cdot LS1 \cdot LS2$$

Nous avons également utilisé une approche par apprentissage pour déterminer la meilleure fonction de score *SA* possible. Contrairement à ce que nous avons fait pour SmartIdent, nous ne nous sommes pas contentés d'apprendre les poids des sous-fonctions dans la fonction empirique, mais nous avons cherché à apprendre la fonction complète. Pour cela nous avons utilisé notre outil de programmation génétique parallèle PyM (voir la section 2.3.1). L'ensemble d'apprentissage était un ensemble *S* de 150 spectres MS/MS pour lesquels nous disposions d'identifications claires. L'ensemble des non-terminaux était les opérateurs arithmétiques classiques (+,-,*,/) et un opérateur conditionnel (siInférieur-alors-sinon). L'ensemble des terminaux était l'ensemble des sous-scores et des coefficients aléatoires. La fonction de fitness *F* à optimiser était :

$$F = \sum_{s \in S} \frac{1}{r^2}$$

avec *r* le rang dans la liste des résultats (triée par ordre décroissant des valeurs de *SA*) du bon peptide de la base de données. On cherche donc la fonction *SA* qui donnerait le rang le plus faible possible à la bonne protéine pour chaque spectre de l'ensemble d'apprentissage. Ainsi, si pour un spectre de l'ensemble d'apprentissage, la fonction de score *SA*, dont on cherche à déterminer la fitness, a donné le meilleur score à la bonne protéine, la contribution à la fitness globale sera de 1. Si, par contre, la fonction *SA* a placé la bonne protéine à un rang plus élevé dans la liste résultat, la contribution à la fitness globale de ce spectre sera inférieure à 1. L'utilisation du programme PyM nous permet d'optimiser plusieurs critères en même temps. Nous cherchons donc également la fonction *SA* la plus compacte possible. Un exemple de fonction *SA* de score *F* optimal est donné dans la figure 22.

Pour valider notre approche, nous avons utilisé le programme Popitam avec la fonction empirique *SE* et avec la meilleure fonction apprise *SA*, ainsi que le programme d'identification MS/MS Mascot pour identifier deux ensembles de spectres de tailles respectives 178 spectres et 505 spectres. Dans chaque situation, nous avons évalué l'efficacité de chaque programme, en comptant le pourcentage des spectres pour lesquels le programme avait placé le bon peptide en première position (rang 1) et en deuxième position (rang 2) dans le cas de Popitam. Dans le cas de Mascot, nous avons compté le pourcentage de spectres pour lesquels Mascot plaçait le bon peptide en première position en lui attribuant un score au-dessus du seuil de significativité et en première position mais au-dessous du seuil de significativité. Ces résultats sont présentés dans la table 6.

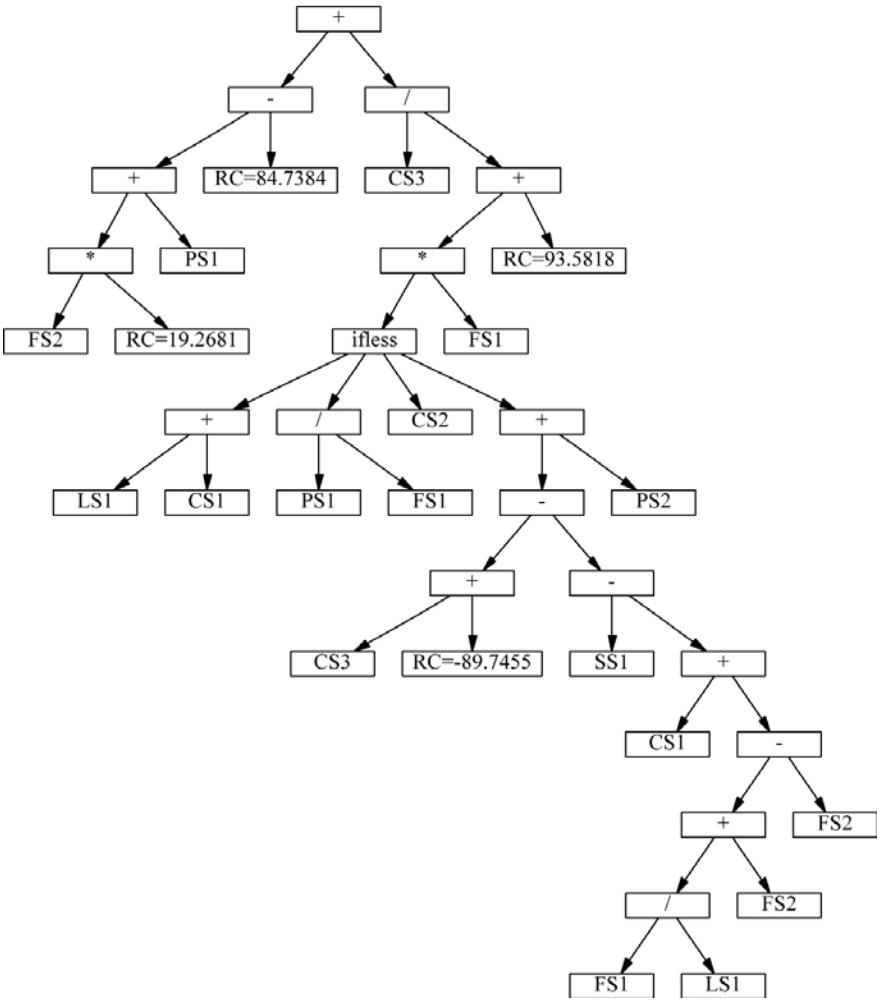


Figure 22. Exemple de fonction de score *SA* découverte par le programme PyM et maximisant la fonction F.

	Popitam avec *SE*		Popitam avec *SA*		Mascot	
	r = 1	*r* = 2	*r* = 1	*r* = 2	> seuil	< seuil
BOL_178 (mammifère)	89.3%	4.5%	93.5%	3.4%	92.1%	5%
NUC_505 (humain)	85.9%	4.4%	95.2%	1.8%	63.1%	3.7%

Table 6. Comparaison de Popitam avec la fonction de score empirique, Popitam avec la fonction de score apprise et Mascot. Les résultats sont présentés en terme de pourcentage de spectres pour lesquels le bon peptide a été placé en première (*r* = 1) et en deuxième (*r* = 2) position pour Popitam et en pourcentage de spectres pour lesquels le bon peptide a été placé en première position avec un score au-dessus (> seuil) et en dessous (< seuil) du seuil de significativité pour Mascot.

3.1.3.2.5 Discussion

Les résultats obtenus montrent plusieurs choses. Tout d'abord, l'apprentissage de la fonction de score est particulièrement efficace car amenant un gain de près de 10% de spectres placés en première position sur le deuxième ensemble de tests. Ensuite, les résultats obtenus par Popitam par rapport à ceux de Mascot sont très encourageants particulièrement pour le deuxième ensemble de tests qui est plus difficile car il contient des spectres de moins bonne qualité. Cela est d'autant plus intéressant que Popitam a été utilisé ici dans une situation pour laquelle il n'a pas été spécialement conçu, c'est-à-dire identifier des peptides non modifiés, ce qui n'est pas le cas de Mascot. Nous avons réalisé avec Popitam des premières expérimentations d'identifications de spectres avec modifications non-connues à l'avance. Les résultats sont prometteurs puisque Popitam a réussi à identifier ces peptides, que les méthodes classiques ne peuvent identifier.

Popitam est cependant sujet à plusieurs limitations. Les spectres qu'il peut identifier doivent contenir des successions de fragmentations suffisamment longues pour pouvoir détecter des tags de longueur minimale. Popitam demande un temps de calcul beaucoup plus important (quelques minutes par spectre) que les autres programmes d'identification MS/MS, du fait, en particulier, de la recherche des cliques. Il n'est donc pas raisonnable d'envisager une utilisation directe dans un contexte de production de données à haut débit. Par contre, une utilisation couplée avec d'autres programmes d'identifications semble tout à fait pertinente, Popitam étant réservé aux identifications qui ont échoué avec les autres programmes, ou dans le cas où l'on suspecterait de possibles modifications.

Plusieurs améliorations sont en cours d'étude, en particulier pour la fonction de score. Nous développons des sous-fonctions de scores plus informatives basées sur les distributions de probabilités des différents types de fragments et des différentes charges des fragments s'inspirant de ceux développés dans (Colinge J. et al., 2003). Nous souhaitons également intégrer une notion de significativité des scores obtenus en analysant les distributions des scores obtenus par les bons peptides et ceux des faux positifs de façon similaire à ce qui a été utilisé dans le scanner moléculaire (voir la section 3.1.2.3).

3.2 Découverte de motifs

3.2.1 Introduction

Les mécanismes de la biologie moléculaire sont basés sur des interactions intra ou inter molécules, principalement des molécules génomiques (ADN et ARN) et des molécules protéiques (protéines et peptides). Si ces mécanismes sont nécessaires, ou tout au moins utiles à la survie de l'organisme, ils vont avoir tendance à être préservés par le processus de l'évolution. Or, ces interactions sont directement liées à la nature des

molécules de bases (nucléotides et acides aminés le plus souvent) qui participent à la liaison. Donc, si les mécanismes sont préservés par l'évolution, les sites d'interactions vont peu changer d'une espèce à l'autre pour conserver leurs propriétés. Un motif biologique est un site participant à une telle interaction.

La découverte de motifs a pour but de localiser des zones, dans des séquences biologiques d'intérêts, susceptibles d'être des motifs biologiques. C'est une étape particulièrement importante pour la compréhension de la fonction des macro-molécules biologiques puisque toute fonction passe par une interaction et donc par un motif biologique. Le principe de la découverte de motif est de détecter, parmi un ensemble de séquences biologiques que l'on suppose impliquées dans un même mécanisme, une ou plusieurs zones dont le niveau de conservation est particulièrement élevé puisque préservé par l'évolution.

Nous nous intéressons à la découverte de deux types de motifs biologiques liés à deux types de conservation. Pour le premier type, la conservation se fait "en l'état", c'est-à-dire que l'on suppose que la conservation s'est faite en autorisant peu de mutations dans le site d'interaction. C'est le type de motif que nous cherchons à découvrir avec l'outil MoDEL de la section 3.2.2. Pour le deuxième type, la conservation s'est faite par co-évolution, c'est-à-dire que les deux sites concernés (au sein de la même molécule en cas d'interactions intra-moléculaires ou entre deux molécules en cas d'interactions inter-moléculaires) ont évolué conjointement de façon à préserver les propriétés d'association de leurs molécules de base. Dans ce cas, la conservation de chacun des deux sites peut être faible mais la co-variation des deux sites élevée. C'est ce type de motifs que nous cherchons à localiser avec l'outil TopMoDEL de la section 3.2.3. Ces deux problèmes sont vus comme des problèmes d'optimisation combinatoire. Le but est d'explorer l'espace de motifs possibles et de détecter celui (ou ceux) qui maximise une fonction mesurant le niveau de conservation. MoDEL et TopMoDEL utilisent des stratégies évolutives couplés à des grimpeurs stricts pour explorer ces espaces.

3.2.2 MoDEL

3.2.2.1 Introduction

La localisation de motifs ou de domaines dans des séquences nucléiques et peptidiques est une tâche centrale de beaucoup de recherches biologiques. L'alignement de tels motifs dans plusieurs séquences peut être à l'origine de la construction de modèles de motifs (séquence consensus, automate, HMM…) utilisés pour rechercher des motifs similaires dans de nouvelles séquences (Attwood T. et al., 2003; Bateman A. et al., 2001; Bucher P., 1990; Sigrist C. et al., 2002) et servir de point de départ à des études sur l'évolution. Dans la plupart des cas, la localisation se fait "à la main" à partir d'outils d'alignements multiples tels que CLUSTALW (Thompson J.D. et al., 1994), T-COFFEE (Notredame C. et al., 2000) or DIALIGN2 (Morgenstern B., 1999), tous basés sur des approches par alignement de séquences deux à deux. Malheureusement, ces approches par alignements deux à deux peuvent ignorer des ressemblances subtiles qui

n'apparaissent que lorsque l'alignement est considéré dans son ensemble (toutes les séquences en même temps) (Notredame C., 2002). D'autres méthodes, cependant, considèrent plusieurs séquences simultanément ou découvrent des mots ou patterns consensus qui s'apparient à toutes ou à un nombre prédéfini de séquences parmi celles dans lesquelles on cherche un alignement. Ces méthodes sont en général capables de détecter des signaux plus faibles (alignements moins bien conservés) mais en contrepartie nécessitent la spécification d'un plus grand nombre de paramètres pour limiter la taille de l'espace des alignements possibles. Elles sont, de ce fait plus, spécifiques que les algorithmes d'alignement multiples classiques.

Nous utiliserons le terme *occurrence* comme dénomination pour tout ensemble de symboles consécutifs présent dans une séquence et nous appellerons *mot* tout ensemble de symboles consécutifs indépendamment de toute séquence. Une classe de ces méthodes découvre des mots avec gaps (c'est-à-dire des mots dans lesquels certains symboles ne sont pas spécifiés) ayant des occurrences exactes dans un nombre prédéfini de séquences. Les principaux sont Pratt (Jonassen I. et al., 1995), Splash (Califano A., 2000) et TEIRESIAS (Rigoutsos I. and Floratos A., 1998) qui sont tous des algorithmes exacts[22]. Une autre classe de méthodes cherche des ensembles d'occurrences, chacune d'elles étant à une distance de Hamming de *k* ou au plus *k* d'un mot unique (a priori inconnu). Ce mot, qui peut n'avoir aucune occurrence exacte dans l'ensemble de séquences, peut être considéré comme l'hypothétique ancêtre commun. Les algorithmes principaux dans cette classe sont PatternBranching (Price A. et al., 2003), Winnover (Pevzner P.A. and Sze S.-H., 2000), Projection (Buhler J. and Tompa M., 2002), Multiprofiler (Keich U. and Pevzner P.A., 2002) et Smile (Marsan L. and Sagot M.-F., 2000). Ce dernier est un algorithme exact qui permet, en plus, de découvrir plusieurs mots reliés par des gaps de tailles variables. D'autres méthodes encore, comme OligoAnalysis (Van Helden J. et al., 1998) et YMF (Sinha S. and Tompa M., 2000) sont basées sur des énumérations exhaustives dans l'espace des mots possibles pour découvrir des mots sur-représentés. Du fait de l'importante quantité de calculs nécessaire pour ces méthodes, leur utilisation est limitée à la découverte de mots courts dans des séquences d'ADN uniquement.

La classe de méthodes qui nous intéresse plus directement est celle qui recherche un ensemble d'occurrences maximalement conservées du point de vue d'une mesure de contenu en information. Ce type de problèmes a été démontré NP-complet (Akutsu T. et al., 2000), donc toutes les méthodes cherchant à le résoudre sont des heuristiques non-exactes. Mis à part CONSENSUS (Hertz G.Z. et al., 1990; Hertz G.Z. and Stormo G.D., 1999), qui contruit ces solutions en suivant une approche gloutonne, toutes les autres méthodes explorent l'espace de recherche par échantillonnage en optimisant une fonction objectif. Font parties de cette catégories, la méthode bien connue du Gibbs Sampler (Lawrence C.E. et al., 1993; Neuwald A.F. et al., 1995), AlingACE (Hughes J.D. et al., 200) et MEME (Bailey T.L. and Elkan C., 1995; Lawrence C.E. and Reilley A., 1990). Notre méthode, MoDEL (Motif Discovery with Evolutionary Learning) (Hernandez D. et al., 2002; Hernandez D. et al., 2004), s'attache à résoudre le problème classique suivant : étant donné un ensemble de séquences fonctionnellement liées, choisir exactement une occurrence par séquence de telle manière que l'ensemble des occurrences soit

[22] Au sens de garantie de découvrir la solution optimale étant donnés les critères choisis.

maximalement similaire. Ce problème est aussi connu sous le nom *d'alignement multiple local sans gap* (*ULMA*). Notre méthode, développée par David Hernandez durant sa thèse, peut être utilisée indifféremment sur tout type d'alphabet de taille raisonnable (ADN, protéine…). MoDEL utilise une stratégie d'exploration hybride consistant en un algorithme génétique (recherche globale) couplé à des grimpeurs spécifiques (recherche locale). Selon la classification de Brazma (Brazma A. et al., 1998), MoDEL est à la fois un algorithme de découverte de motifs guidé par la séquence et par le motif car il s'appuie sur deux représentations (deux espaces de recherches sont explorés) : premièrement, l'espace de tous les mots d'une longueur donnée sur un alphabet donné et deuxièmement, l'espace de tous les ULMAs d'une longueur donnée. L'idée clé est d'utiliser les résultats de la recherche globale dans l'espace des mots pour limiter l'exploration de l'espace des ULMAs aux régions les plus prometteuses.

3.2.2.2 Définitions et méthodes

3.2.2.2.1 Définitions générales

Soit $S = \{S_1, S_2, \ldots, S_N\}$ un ensemble de N séquences de longueur T[23]. $S_{i,j}$ est le $j^{ième}$ symbole de la $i^{ième}$ séquence. Toutes les séquences de S sont construites sur un alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$ avec $K = |\Sigma|$ (4 pour le génomique et 20 pour les protéines). Soit $M = \{m_1, m_2, \ldots, m_W\}$ un mot de longueur W défini sur Σ. L'ensemble de tous les mots possibles est l'espace $\mathcal{M}$ de taille K^W. Soit $P = \{p_1, p_2, \ldots, p_N\}$un vecteur de position avec p_i le début d'une occurrence dans la séquence S_i de longueur W. P représente donc un ULMA comportant exactement une occurrence dans chaque séquence. L'ensemble de tous les ULMAs P possibles définit l'espace de recherche $\mathcal{P}$ et sa taille est $(T - W + 1)^N$. On appelle un ULMA particulier P, un point de $\mathcal{P}$. Le but de notre algorithme est de découvrir le point de $\mathcal{P}$ qui maximise une fonction objectif qui représente le niveau de conservation des occurrences dans P.

3.2.2.2.2 La fonction objectif

Le contenu en information se mesure par la divergence de Kullback-Leiber (Kulback S., 1968), appelée aussi entropie relative. Cette mesure exprime à quel point les fréquences observées dans le ULMA sont exceptionnelles étant données les fréquences du bruit de fond. On construit, à partir de l'ULMA, une table des fréquences observées F telle que pour $k = 1, \ldots, K$ et $j = 0, \ldots, W-1$ on ait :

$$F_{k,j} = \frac{1}{N} \cdot \sum_{i=1}^{N} J(k, S_{i,p_i+j})$$

avec

[23] Pour simplifier les notations, nous considérons que toutes les séquences ont la même longueur, mais MoDEL fonctionne avec n'importe quelle combinaison de longueur de séquences.

$$J(k,a) = \begin{cases} 1 & si\ a = \sigma_k \\ 0 & si\ a \neq \sigma_k \end{cases}$$

La valeur de fitness (notre fonction objectif) I(*P*) d'un vecteur *P* est donnée par la valeur d'entropie relative de *F* :

$$I(P) = \sum_{i=1}^{K} \sum_{j=1}^{W} F_{i,j} \log\left(\frac{F_{i,j}}{F_i^0}\right)$$

avec F^0 l'estimation des fréquences du background observées sur l'ensemble des séquences entières. F^0 est estimée au début de l'algorithme et est conservée inchangée pendant toute l'exploration. L'entropie relative est directement liée au ratio du *logarithme de la vraisemblance* (log-likelihood ratio) (Bailey T.L., 1993). Ces deux fonctions objectif ont déjà été utilisées pour mesurer le niveau de conservation d'un ULMA (Hertz G.Z. et al., 1990; Kulback S., 1968; Lawrence C.E. et al., 1993; Lawrence C.E. and Reilley A., 1990) et la valeur du sens biologique que l'on peut leur attribuer a été clairement montrée.

3.2.2.3 Les opérateurs de projection

Le processus d'exploration de MoDEL tire parti de la double représentation en mot et en ULMA des motifs possibles, les mots étant considérés comme des hypothétiques ancêtres communs des ULMAs. Pour cela, il faut pouvoir passer d'une représentation à l'autre. Nous avons défini deux opérateurs de projections d'un espace vers l'autre. Le premier opérateur (MtoP) projette un point de $\mathcal{M}$ vers $\mathcal{P}$, tandis que le second (PtoM) projette un point de $\mathcal{P}$ vers $\mathcal{M}$.(voir figure 23).

L'opérateur MtoP requiert un alignement d'un mot *M* avec toutes les séquences de S. Pour $i = 1, \ldots, N$ et $j = 1, \ldots, T - W + 1$, on détermine les valeurs p_i du vecteur de l'ULMA *P* par :

$$p_i = \arg\max_j \sum_{k=1}^{W} J(S_{i,j+k-1}, m_k)$$

avec

$$J(a,b) = \begin{cases} 1 & si\ a = b \\ 0 & si\ a \neq b \end{cases}$$

Il en découle que p_i correspond à la position dans S_i pour laquelle le mot *M* et son occurrence dans S_i sont à une distance de Hamming minimale l'un de l'autre. Cette procédure conduit souvent à des ULMAs avec plusieurs positions ambiguës pour les p_i, ce qui signifie que plusieurs occurrences de valeur maximale ont été découvertes dans une même séquence. Nous levons ces ambiguïtés à l'aide d'un algorithme glouton dont le

principe est le suivant : toutes les séquences comportant des occurrences non ambiguës sont sélectionnées pour former un noyau sur lequel est calculé le score de contenu en information I(*P*). Puis les autres séquences sont rajoutées une à une, en choisissant à chaque fois l'occurrence qui, ajoutée aux occurrences du noyau, maximise la nouvelle valeur de I(*P*).

L'opérateur de projection a une propriété particulièrement intéressante : grâce aux alignements réalisés dans chaque séquence, les occurrences résultant de la projection présentent une similarité très supérieure à celle d'un ensemble d'occurrences prises au hasard (un point quelconque de $\mathcal{P}$). Cette propriété permet de débuter l'exploration dans $\mathcal{P}$ à partir de points de haute valeur de fitness. Nous appelons Q, l'ensemble des points de $\mathcal{P}$ qui sont atteignables par une projection d'un point de $\mathcal{M}$. :

$$Q = \{ p_i \in \mathcal{P} \mid \forall\ m_i \in \mathcal{M}, p_i = \mathrm{MtoP}(m_i)\}$$

Par construction, cet espace est donc au plus de la taille de $\mathcal{M}$..

Nous définissions un opérateur complémentaire, PtoM, qui produit un mot *M* à partir d'un ULMA *P*. Pour cela, nous choisissons le mot le plus probable en fonction de la matrice de fréquences *F*. Plus formellement, pour $i = 1, \ldots, K$ et $j = 1, \ldots, W$:

$$m_j = \sigma_k \quad avec\ k = \arg\max_i (F_{i,j})$$

Si plusieurs k sont possibles pour un j donné, l'un d'eux est choisi aléatoirement.

On peut remarquer que la correspondance entre les espaces $\mathcal{M}$ et $\mathcal{P}$, réalisée par les opérateurs MtoP et PtoM, n'est pas symétrique. En effet, un mot projeté de $\mathcal{M}$ vers $\mathcal{P}$ et projeté de nouveau vers $\mathcal{M}$ ne redonne pas forcement le mot initial. Cela est également vrai si on part d'un ULMA de $\mathcal{P}$.

3.2.2.4 Exploration des espaces

MoDEL combine deux stratégies d'exploration. La première réalise une recherche globale dans $\mathcal{P}$ en explorant Q à l'aide d'un algorithme génétique. La deuxième consiste en une recherche locale dans $\mathcal{P}$ en appliquant des grimpeurs à partir des points de Q sélectionnés par la première stratégie.

3.2.2.4.1 L'algorithme génétique

Nous utilisons un algorithme génétique classique pour explorer l'espace des mots $\mathcal{M}$.. Un chromosome de la population est un vecteur de longueur *W* où chaque variable prend ses valeurs dans Σ. De manière à préserver un haut niveau de diversité dans la population, nous avons choisi un opérateur de sélection par tournoi de taille 2. La fonction de fitness est la valeur de contenu en information de la projection dans $\mathcal{P}$ des mots de la population. Nous avons développé une série d'opérateurs génétiques dédiés a

notre application. Chacun d'eux est basé sur une fonction de voisinage adaptée à explorer l'espace des mots pour découvrir des points de Q intéressants.

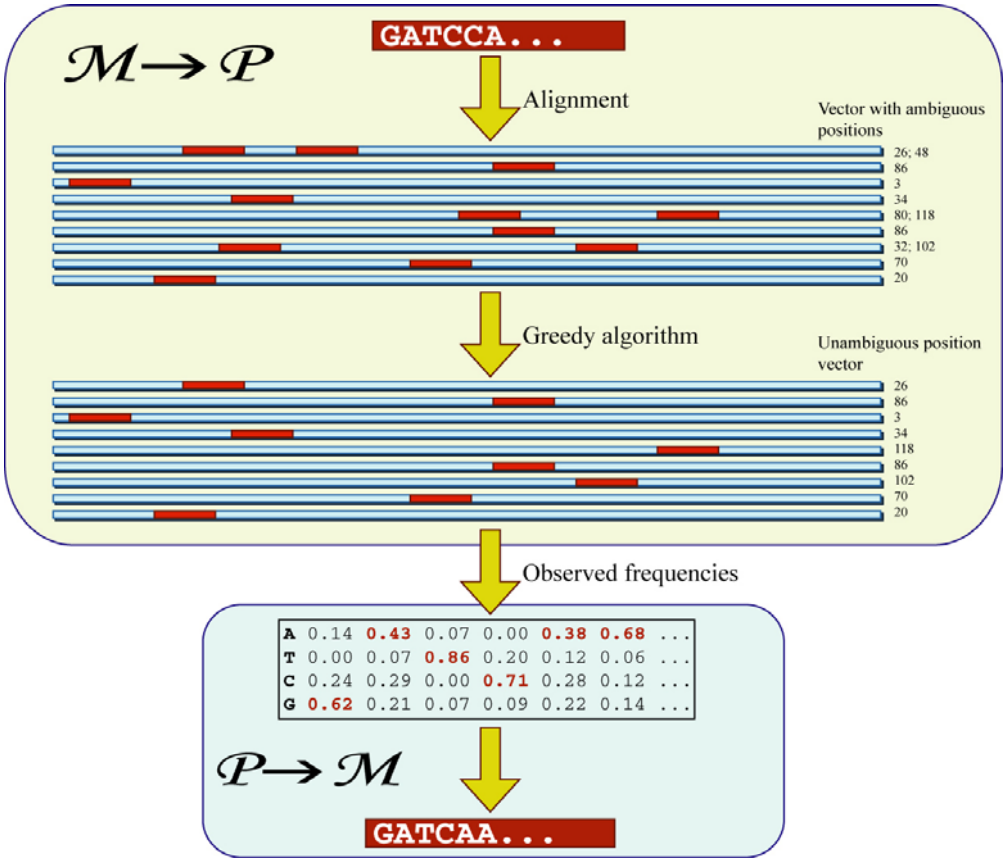


Figure 23. Opérateurs de projection permettant de déterminer le correspondant d'un point, d'un espace de recherche dans l'autre espace de recherche. $\mathcal{M} \rightarrow \mathcal{P}$: un mot est aligné dans chaque séquence de façon à découvrir les positions qui minimisent la distance de Hamming. Un algorithme glouton permet de lever les ambiguïtés lorsque plusieurs positions sont maximales dans une même séquence. $\mathcal{P} \rightarrow \mathcal{M}$: à partir de la matrice de fréquence F un mot de probabilité maximale est généré.

3.2.2.4.2 Les opérateurs génétiques

Nous avons conçu sept opérateurs génétiques dont la plupart est spécifique à notre application. Chacun d'eux est appliqué, avec une probabilité prédéfinie, aux chromosomes sélectionnés dans l'ancienne population pour constituer de nouveaux chromosomes qui sont placés dans la nouvelle population. Nous avons trois opérateurs de crossover, un opérateur de glissement, un opérateur de mutation et deux opérateurs spéciaux générant de nouveaux points de $\mathcal{M}$ en passant par une projection dans $\mathcal{P}$. Ces deux derniers opérateurs seront décrits dans la section suivante.

Les crossovers ont pour objectif de combiner des portions de solutions (building blocks) corrélés de façon à former de nouvelles solutions de plus haute valeur de fitness. Dans notre cas, les building blocks seront des sous-mots issus des mots de la population. Nous utilisons trois opérateurs de crossover différents, chacun combinant deux chromosomes. 1) Le crossover classique à un point (*OPC*) : une position est tirée aléatoirement (la même dans chaque chromosome) et tous les symboles des deux chromosomes situés après cette position sont échangés. 2) Le crossover uniforme (*PPC*) : chaque symbole d'un chromosome est échangé, avec une certaine probabilité, avec le symbole correspondant de l'autre chromosome. 3) le crossover par alignement (*SC*, voir figure 24) : ce crossover a été conçu pour échanger une information maximalement significative entre les deux chromosomes. En effet, il arrive fréquemment que deux mots (deux chromosomes) aient la même occurrence dans une ou plusieurs séquences, à l'exception d'un décalage d'un ou plusieurs symboles. Dans ce cas, les crossover PPC et SC vont être destructeurs puisqu'ils vont échanger de l'information n'ayant pas la même signification, formant deux nouveaux mots très différents. Pour éviter ce phénomène, l'opérateur SC va réaliser préalablement un alignement entre les deux mots de façon à déterminer les parties les plus similaires. Un score de similarité est calculé de la façon suivante : étant donnés $M^a = m_1^a, \ldots, m_W^a$ et $M^b = m_1^b, \ldots, m_W^b$, deux mots de longueur *W*, il existe $2 \cdot W$-1 alignements possibles. Le score de l'alignement est donné par le nombre de symboles identiques à position égale de l'alignement, divisé par la longueur totale de l'alignement. Les échanges sont réalisés de la même manière qu'avec l'opérateur PPC, mais uniquement dans la région de recouvrement correspondant à l'alignement de score maximal de chaque chromosome.

Deux opérateurs s'appliquant à un seul chromosome à la fois sont aussi utilisés. Un opérateur de mutation (*MO*) change aléatoirement le symbole contenu à une ou plusieurs positions du chromosome. Un opérateur de glissement (*SO*, voir figure 25) déplace tous les symboles d'une position vers la droite ou la gauche. Le symbole "éjecté" est éliminé alors qu'un nouveau symbole choisi aléatoirement est inséré à la position libérée.

3.2.2.4.3 Exploration locale dans $\mathcal{P}$

L'exploration effectuée uniquement par l'algorithme génétique n'est pas suffisante. En effet, le maximum global peut tout à fait être situé dans une zone de $\mathcal{P}$ non incluse dans Q (il a même très peu de chance de se situer dans Q). Une recherche locale dans $\mathcal{P}$ est alors nécessaire pour accéder à de nouveaux points et espérer découvrir le maximum global. Après chaque itération de l'algorithme génétique, environ 5% des nouvelles solutions créées par les opérateurs génétiques sont choisies par une sélection par tournoi pour être optimisées par une recherche locale dans $\mathcal{P}$. Cette recherche locale est réalisée à l'aide de deux opérateurs grimpeurs (voir figure 26) : un opérateur d'*optimisation par voisinage de séquence* (OVS) et un opérateur de *correction de phase* (CP) (Lawrence C.E. et al., 1993).

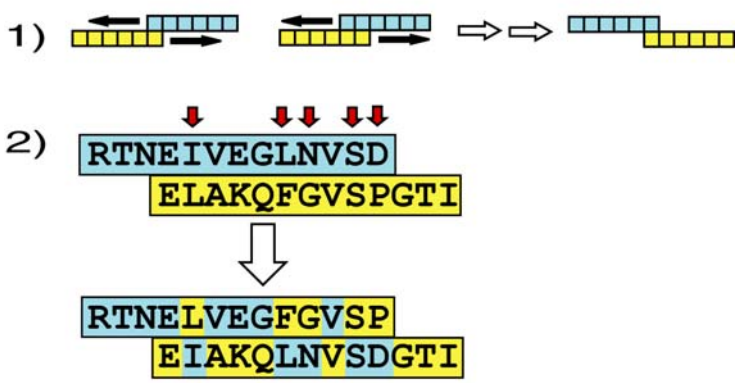


Figure 24. Opérateur de crossover par alignement. 1) Deux chromosomes (mots) sont sélectionnés dans la population et alignés jusqu'à obtention d'une position maximisant une mesure de ressemblance entre les parties appariées. 2) Crossover uniforme entre les deux segments alignés.

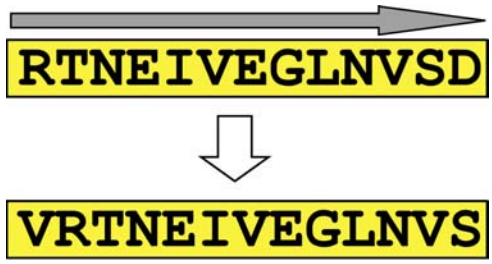


Figure 25. Opérateur de glissement. Tous les symboles du mot sont décalés d'une position vers la droite ou vers la gauche (ici à droite). Le symbole "éjecté" (ici le D) est perdu. Un symbole aléatoire (ici le V) est ajouté à la position libérée.

L'opérateur d'optimisation par voisinage de séquence fonctionne comme suit : étant donné un ULMA $P = p_1, p_2, \ldots, p_N$, une seule dimension p_i est modifiée à la fois, les autres étant conservées inchangées. Cette dimension est échantillonnée sur toutes les positions possibles (correspondant aux $T - W + 1$ facteurs de S_i), et p_i prend alors la valeur de la position qui maximise le contenu en information I(P) de cette occurrence avec toutes celles qui n'ont pas été changées. Toutes les dimensions sont modifiées les unes après les autres, soit dans un ordre prédéfini, soit par tirage aléatoire sans remise. Le processus complet est répété jusqu'à convergence, ce qui requiert en général moins de 10 itérations. On pourra remarquer que les déplacements dans l'espace des alignements réalisés par l'opérateur d'optimisation par voisinage de séquences sont très similaires à

ceux réalisés par l'algorithme de Gibbs Site Sampler (Lawrence C.E. et al., 1993). La différence provient du fait que le Gibbs Site Sampler utilise un grimpeur stochastique à partir de points aléatoires de $\mathcal{P}$. Ce grimpeur procède par choix biaisé de déplacement dans le voisinage, chaque voisin ayant une probabilité d'être sélectionné proportionnelle à la probabilité que l'occurrence correspondante ait été générée par le modèle *F* actuel. La justification de l'utilisation d'un grimpeur stochastique est de pouvoir échapper aux maximums locaux. Dans MoDEL, les déplacements sont déterministes dans le sens où à chaque étape c'est le voisin de meilleur fitness qui est choisi. La diversité de la recherche est assurée par l'utilisation de l'algorithme génétique qui produit des points de $\mathcal{Q}$ pertinents vis-à-vis de la fonction à optimiser, les opérateurs grimpeurs n'étant utilisés que dans un espace restreint autour des points de $\mathcal{Q}$ sélectionnés.

L'opérateur de correction de phase consiste simplement à déplacer toutes les valeurs des p_i simultanément, d'une ou plusieurs positions vers la gauche ou la droite. Cela permet d'assurer que l'alignement découvert n'est pas décalé dans son ensemble de quelques positions par rapport à l'alignement optimal.

Ces optimisations par grimpeurs ne sont pas effectuées sur la population complète de l'algorithme génétique mais uniquement sur une petite partie (environ 5%, sélectionnée en fonction de sa fitness) et cela pour deux raisons : les optimisations par grimpeurs sont des processus coûteux en temps de calcul[24] et l'expérience a montré que de nombreux points de $\mathcal{Q}$ mènent, après utilisation de grimpeurs stricts, à une convergence vers une solution unique (ces points font donc partie du bassin d'attraction d'une même solution). De ce fait, l'utilisation de ces opérateurs sur toutes les solutions serait à la fois coûteuse en temps de calcul et peu productive en terme de nouvelles solutions produites.

Les deux derniers opérateurs, cités dans la section précédente, sont basés sur les opérateurs de recherche locale dans $\mathcal{P}$. Comme les autres opérateurs, ils provoquent des mouvements de $\mathcal{M}$ vers $\mathcal{M}$ mais ces mouvements se font à travers une projection dans $\mathcal{P}$. Le premier (*SNtoM*) est l'opérateur d'optimisation par voisinage de séquence. Un mot est projeté dans $\mathcal{P}$, ensuite l'ULMA résultant est optimisé par le grimpeur et projeté de nouveau dans $\mathcal{M}$. Le second (*PStoM*) suit le même principe mais utilise à la place l'opérateur de correction de phase. Ces deux opérateurs sont déterministes et produisent un mouvement dans $\mathcal{M}$ guidé par l'optimisation dans $\mathcal{P}$.

[24] L'optimisation par grimpeur de 5% de la population requiert environ 50% du temps CPU total du programme MoDEL.

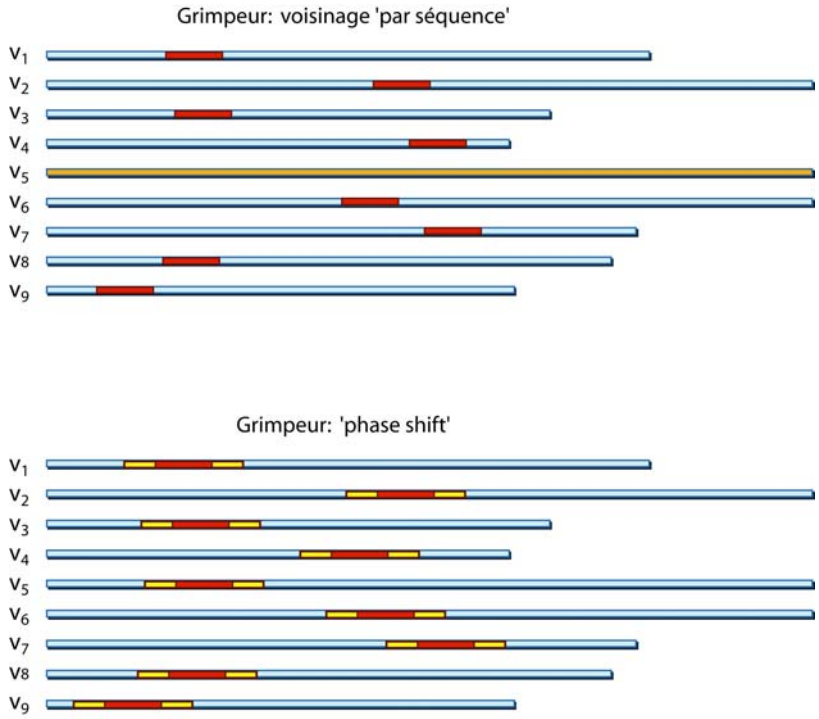


Figure 26. Voisinage pour l'opérateur d'optimisation par voisinage de séquence (en haut) et pour l'opérateur de correction de phase (en bas). Dans le premier cas, la séquence V_5 a été sélectionnée, le voisinage de l'alignement actuel consiste donc en l'ensemble des positions possibles sur la séquence V_5. Dans le deuxième cas, le voisinage correspond à l'ensemble des alignements décalés simultanément dans toutes les séquences d'une ou plusieurs positions à droite ou à gauche de l'alignement rouge.

3.2.2.5 Schéma global de MoDEL et complexité temporelle

Plusieurs méthodes d'optimisation fonctionnent conjointement dans MoDEL. La figure 27 donne une vision globale du fonctionnement de MoDEL. Bien que l'algorithme soit stochastique, nous pouvons donner une estimation de sa complexité temporelle globale. Si le nombre de solutions simultanément échantillonnées (la taille de la population) est considéré constant, la complexité temporelle de MoDEL est de l'ordre de O(NTWG), avec N le nombre de séquences, T la longueur moyenne des séquences, W la longueur de l'alignement cherché et G le nombre d'itérations. Nous donnons dans la section 3.2.3 des résultats de comparaisons de MoDEL avec d'autres programmes d'alignements multiples locaux sans gap qui permettent de se faire une idée plus précise du temps d'exécution effectif de MoDEL sur des problèmes réels.

3.2.2.6 Résultats

Nous comparons, dans cette section, plusieurs stratégies d'optimisation dédiées au problème de l'alignement multiple local sans gap. Les comparaisons sont effectuées sur

deux jeux de données différentes : un ensemble de séquences protéiques contenant un motif structurel hélice-tour-hélice (HTH) et un ensemble de séquences aléatoires. Nous proposons finalement une discussion sur le challenge du motif caché, énoncé par Pevzner (Pevzner P.A. and Sze S.-H., 2000), en tant que problème de référence pour la comparaison des algorithmes de découverte de motifs.

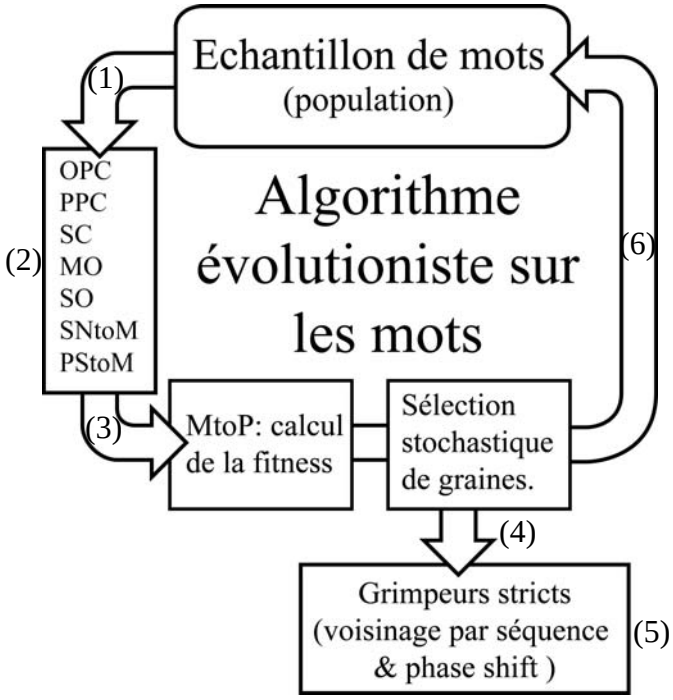


Figure 27. Processus global de MoDEL. (1) Sélection par tournoi dans laquelle certaines solutions sont choisies stochastiquement en fonction de leur fitness. (2) Les solutions sélectionnées sont stochastiquement transformées par l'application d'une série d'opérateurs. (3) Les nouvelles solutions sont projetées dans Q par l'intermédiaire de l'opérateur MtoP et leurs fitness sont calculées. (4) Sélection par tournoi de 5% de la nouvelle population. (5) Utilisation des opérateurs de recherche locale dans $\mathcal{P}$ sur ces solutions sélectionnées et mémorisation des meilleurs alignements. (6) Les nouvelles solutions remplacent les anciennes dans la population.

3.2.2.6.1 La découverte du motif HTH

Nous avons constitué un jeu de données de séquences protéiques contenant un motif commun (le motif HTH) de telle sorte que les séquences choisies soient maximalement divergentes. Le motif étant annoté dans chacune de ces séquences, on dispose ainsi d'un problème difficile pour lequel la solution supposée optimale est connue. Le but de cette comparaison n'est pas de démontrer la capacité de la fonction de contenu en information à représenter la similarité de séquences biologiques, mais d'évaluer les capacités d'exploration d'un espace de recherche, guidé par l'optimisation d'une même fonction objectif, de différentes stratégies de découverte de motifs. Ces

comparaisons se font en terme de qualité de l'alignement obtenu à temps égal d'exploration. Il est bien évident que pour des raisons d'équité de la comparaison, les méthodes comparées sont uniquement celles qui ont été spécifiquement conçues pour résoudre le problème de l'alignement multiple local sans gap et qu'elles cherchent toutes à optimiser des fonctions objectifs équivalentes.

Le jeu de données HTH

Le motif structural HTH est présent dans une large variété de protéines de liaison avec l'ADN existant dans sur l'ensemble du règne vivant. Environ 23 lignées évolutives (clades) sont connues (Rosinski J.A. and Atchley W.R., 1999). Au sein de chaque clade les protéines sont très similaires. Il existe cependant une divergence extrêmement importante entre les séquences de clades différentes. L'étude évolutive menée par Rosinski et Atchley suggère que tous les motifs HTH ont évolué à partir d'un ancêtre commun. Ainsi, bien que notre jeu de données contienne 16 séquences provenant de 16 clades différentes (pour rendre le problème de l'alignement le plus difficile possible du fait du très faible niveau de conservation), il est raisonnable de penser pouvoir découvrir le motif HTH par similarité. Les 16 séquences sélectionnées ont une longueur moyenne de 217 acides aminés (minimum 61 et maximum 613). L'alignement de plus haut score que nous ayons trouvé, toutes méthodes confondues (voir la figure 28), a une valeur d'entropie relative de 32.71 pour une longueur fixée de 20 acides aminés. Comme cet alignement correspond exactement aux annotations sur la position réelle de sites HTH dans ces séquences, nous avons de bonnes raisons de penser que cette valeur correspond au maximum global sur ces séquences. Nous prenons donc cet ULMA comme référence de la réussite de l'exploration.

Il est important de préciser que les occurrences de HTH considérées sont trop faiblement conservées pour pouvoir être retrouvées avec des approches comme Smile et Winnover, qui recherchent un alignement dont toutes les occurrences sont à une distance de Hamming d'au plus k par rapport à un mot externe unique. Tous les programmes classiques d'alignement multiple, tels que CLUSTALW ou T-Coffee, fondés sur la comparaison de séquences deux à deux, sont également incapables d'aligner la plupart des sites HTH. En effet, le niveau de conservation ne devient supérieur à celui de séquences prises au hasard que lorsque l'on considère simultanément toutes les occurrences du ULMA, comme le fait le score de contenu en information.

Retrouver l'ULMA de référence

Nous comparons tout d'abord la capacité de plusieurs programmes à retrouver l'ULMA de référence. Cette comparaison porte, en plus de MoDEL, sur trois programmes très utilisés : le Gibbs Site Sampler[25] (Lawrence C.E. et al., 1993), MEME

[25] Nous utilisons plus précisément l'implémentation du MotifSampler(Neuwald A.F. et al., 1995), disponible sur le site ftp://ncbi.nlm.nih.gov, en utilisant les paramètres spécifiques correspondant au Site Sampler.

(Bailey T.L. and Elkan C., 1994) et CONSENSUS (Hertz G.Z. and Stormo G.D., 1999). Tous ces programmes optimisent la même fonction objectif ou une fonction équivalente[26] et peuvent prendre en compte les mêmes contraintes (taille de l'alignement fixée et rechercher exactement une occurrence par séquence). Les résultats montrent que MEME et CONSENSUS échouent à découvrir l'alignement de référence et restent bloqués sur des maximums locaux. La table 7 donne les scores obtenus par les quatre programmes. Pour que les tests soient les plus équitables possible, nous avons testé un grand nombre de combinaisons des paramètres de ces programmes de manière à déterminer ceux qui leur permettaient d'atteindre les meilleurs résultats. Une valeur de similarité entre l'ULMA découvert et l'ULMA de référence est indiquée dans la table. Elle représente le pourcentage de paires de symboles identiques entre les deux ULMAs. Les algorithmes de MoDEL et du Gibbs Site Sampler étant stochastiques, les résultats présentés correspondent au temps minimum nécessaire pour retrouver l'ULMA de référence avec une probabilité d'au moins 0.95 (voir section suivante).

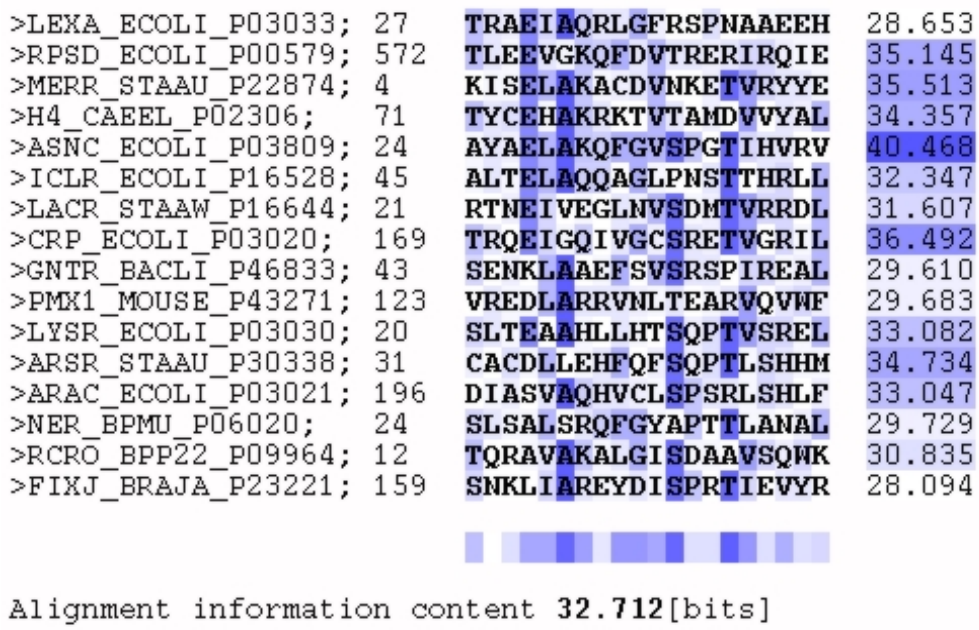


Figure 28. Le domaine HTH sur les 16 séquences sélectionnées : le ULMA de référence. Le nombre situé à gauche de l'alignement correspond à la position de l'occurrence dans la séquence et le nombre de droite correspond à une mesure basée sur le logarithme du ratio de vraisemblance des décomptes observés sachant *F* par rapport aux décomptes observés sachant le bruit de fond (le table de fréquence *F* dérivée de l'alignement). Les couleurs représentent une mesure de la participation de chaque caractère au score de sa colonne. L'image est issue de l'interface du programme MoDEL disponible sur le serveur : http://idefix.univ-rennes1.fr:8080/PatternDiscovery.

[26] Le Gibbs Site Sampler optimise une fonction légèrement différente, utilisant des valeurs de fréquence du background non-fixées. Après avoir réalisé des tests avec MoDEL utilisant la fonction du GibbS Site Sampler, nous pouvons garantir que cette différence de score n'a pas de conséquence sur la capacité du programme à découvrir l'alignement de référence.

	RE	Sim	tCalc
MoDEL	32.71	1.00	4
Site Sampler*	32.71	1.00	12
CONSENSUS	31.96	0.01	5
CONSENSUS*	32.42	0.65	65
MEME	28.38	0.41	1

Table 7. Comparaison de plusieurs méthodes pour la découverte du motif HTH dans un ensemble de 16 séquences maximalement distantes. L'étoile (*) signifie que le programme a été exécuté avec des paramètres optimisés pour ce problème. Pour MEME, les paramètres par défaut étaient les paramètres optimaux. Une valeur de similarité entre les ULMAs découverts et l'ULMA de référence est donnée dans la colonne Sim. tCal est le temps CPU en secondes (Athlon 1.6 Ghz) en moyenne pour 100 exécutions. Les seules méthodes capables de retrouver l'ULMA de référence sont le Gibbs Site Sampler et MoDEL. MEME et CONSENSUS, étant des méthodes déterministes, convergent systématiquement vers un maximum local.

Comparaison de l'optimisation faite par MoDEL avec des stratégies plus simples

Il est important de se demander à quel point la stratégie d'exploration complexe de MoDEL apporte un avantage significatif par rapport à une exploration consistant uniquement en de multiples processus d'exploration simples par grimpeur. Pour illustrer l'importance de la recherche globale par l'algorithme génétique ainsi que celle de l'opérateur de projection MtoP, nous avons comparé, à temps CPU identique, les résultats de MoDEL à ceux obtenus par des grimpeurs à départs multiples en partant de points aléatoires. Le premier de ces grimpeurs (stratégie P) consiste simplement en l'utilisation des opérateurs OVS et CP à partir de points aléatoires de $\mathcal{P}$. Le deuxième (stratégie Q) fonctionne de façon similaire mais à partir de points aléatoires de $\mathcal{Q}$, construits par projection, avec l'opérateur MtoP, de mots aléatoires. Le dernier est l'opérateur du Site Sampler (stratégie Gibbs), que l'on peut voir comme un grimpeur stochastique à partir de points aléatoires de $\mathcal{P}$. La figure 29 montre le taux de succès (pourcentage des 100 exécutions indépendantes ayant permis de découvrir l'ULMA de référence) en fonction du temps CPU pour les quatre stratégies (avec la stratégie AG correspondant à l'algorithme MoDEL). Les résultats permettent de constater une nette amélioration des performances des stratégies P, Q et AG par rapport à la stratégie Gibbs. Les trois autres stratégies présentent des performances très similaires. Nous donnons également les résultats en terme de nombre de points de $\mathcal{P}$ pour lesquels la fonction objectif a été évaluée. Les trois mêmes stratégies sont de nouveau nettement meilleures que la stratégie Gibbs. De plus, on remarque que la stratégie de MoDEL permet de diminuer significativement le nombre de points de l'espace de recherche évalués, ce qui confirme l'efficacité de la sélection des mots par l'algorithme génétique pour réduire l'exploration. La fonction objectif que nous utilisons ici, le contenu en information, est relativement peu coûteuse à évaluer, ce qui fait que le gain en nombre d'évaluations de l'algorithme

génétique compense tout juste le temps nécessaire à l'algorithme génétique lui-même. Dans le cas d'une fonction objectif demandant plus de temps de calcul, par exemple une mesure de contenu en information qui prendrait en compte les caractères ambigus ou la similarité entre acides aminés, l'intérêt de réduire le nombre de points de l'espace de recherche évalués serait beaucoup plus important et permettrait un gain significatif dans le temps d'exécution globale du programme.

3.2.2.6.2 Jeu de données artificielles

Dans cette section, nous présentons une autre comparaison entre MoDEL et le Gibbs Site Sampler, les autres résultats ayant par ailleurs montré la supériorité du Gibbs Site Sampler sur MEME et CONSENSUS dans le cas du problème avec une occurrence exactement par séquence, résultats confirmant ceux de (Akutsu T. et al., 2000). Réaliser une comparaison à partir de données artificielles n'est pas trivial car, à notre connaissance, il n'existe pas de méthode pour générer un jeu de séquences contenant un ULMA avec un contenu en information donné juste supérieur au niveau du bruit (condition nécessaire pour que le problème soit difficile). Nous avons choisi de nous placer dans un contexte purement d'optimisation combinatoire et de comparer uniquement la capacité de chacune des méthodes à explorer un espace de recherche difficile pour découvrir le point de valeur maximale. Pour cela, nous avons constitué un jeu de séquences aléatoires sur un alphabet de taille 4 (pour l'ADN), chaque symbole ayant la même probabilité d'occurrence à chaque position. Il est bien évident que dans de telles séquences, tout ULMA découvert ne peut pas être statistiquement significatif mais cela ne retire rien à l'intérêt de pouvoir découvrir un point de valeur maximale dans un grand espace de recherche.

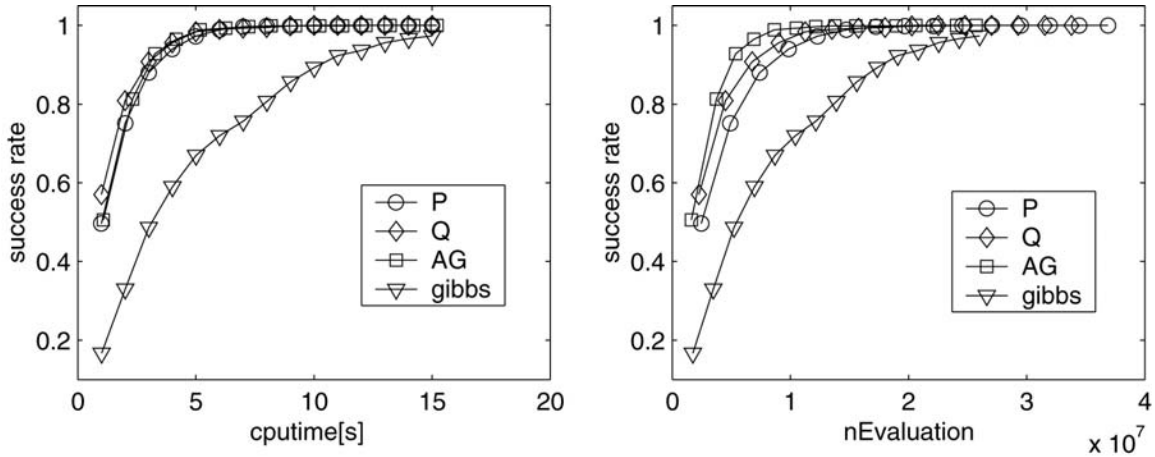


Figure 29. Comparaison des performances de la stratégie de MoDEL (AG) avec des stratégies plus simples sur le problème du HTH. La stratégie P applique les opérateurs OVS et CP à partir de points aléatoires de $\mathcal{P}$. La stratégie Q fait de même à partir de points aléatoires de Q. La stratégie gibbs est celle du Gibbs Site Sampler. Dans la figure de droite, le taux de succès de chaque méthode est présenté en fonction du temps CPU (Athlon 1.6 Ghz). Dans la figure de gauche, le taux de succès est présenté en fonction du nombre de points de l'espace de recherche évalués.

Nous avons généré trois séries de données (A, B et C). Chaque série est composée de 500 ensembles de séquences de longueur 1000. Les ensembles de la série A sont composés de 20 séquences, ceux de la série B de 25 séquences et ceux de la série C de 30 séquences. Pour que les résultats soient comparables, les deux méthodes doivent optimiser la même fonction objectif. Nous avons implémenté la fonction objectif du Gibbs Site Sampler, légèrement différente de la nôtre car considérant un background variable, dans MoDEL. Les paramètres d'exploration ont été ajustés pour que les deux méthodes disposent du même temps CPU et utilisent les mêmes contraintes (longueur de l'alignement et recherche d'exactement une occurrence par séquence). Ils ont également été optimisés pour que chaque méthode soit maximalement efficace sur ces données. La table 8 présente les résultats des deux méthodes sur les trois jeux de données. Elle fait apparaître, pour chaque série, la fréquence avec laquelle chaque méthode fait mieux que l'autre. Même si les scores obtenus pour chaque méthode sont différents, ils peuvent correspondre à des alignements presque identiques. Pour mesurer la différence entre les solutions obtenues, nous avons utilisé le coefficient de performance (PC) défini par Pevzner (Pevzner P.A. and Sze S.-H., 2000). Soit K_1 l'ensemble des positions du premier ULMA et K_2 l'ensemble des positions du deuxième, on définit PC par :

$$PC = \frac{|K_1 \cap K_2|}{|K_1 \cup K_2|}$$

Sur la série A, même si MoDEL découvre de meilleurs alignements un plus grand nombre de fois, les alignements sont relativement similaires (environ 70% des occurrences découvertes sont les mêmes pour les deux méthodes). Dans les séries B et C, par contre, MoDEL obtient des résultats bien meilleurs que le Gibbs Site Sampler. Cela peut s'expliquer par le fait que la taille de l'espace de recherche augmente exponentiellement avec le nombre de séquences et que la stratégie d'exploration devient de plus en plus prépondérante pour garantir la qualité des solutions obtenues.

	M>G	M=G	M<G	SA	tM	tG
A	0.42	0.48	0.10	0.70	412	374
B	0.90	0.05	0.05	0.47	529	636
C	0.99	0.00	0.01	0.35	661	762

Table 8. Comparaison des capacités d'exploration de MoDEL et du Gibbs Site Sampler sur trois séries (A, B et C) d'ensembles de séquences aléatoires. M > G, M = G et M < G signifient respectivement la fréquence des exécutions dans lesquelles MoDEL trouve un alignement de plus haute valeur que le Gibbs Site Sampler, de valeur égale et de valeur inférieure. Une mesure de similarité entre les alignements trouvés par les deux méthodes est donnée dans la colonne SA. tM et tG correspondent respectivement aux temps CPU moyen en secondes pour MoDEL et Gibbs Site Sampler (Pentium IV 1.5 GhZ).

3.2.3.3 Discussions et résultats sur le challenge du motif caché

Nous discutons dans cette section de la validité du problème challenge proposé par Pevzner (Pevzner P.A. and Sze S.-H., 2000) en tant que problème de référence pour la comparaison des différentes méthodes de découverte de motifs. Ce problème consiste à découvrir un motif dans des séquences d'ADN artificielles générées par le modèle FM (nombre de mutations fixe). Soit M un mot de longueur fixée l. Créer N nouveaux mots M_1, M_2, …, M_N, chacun étant une transformation du mot initial M en effectuant k substitutions de symboles. Ces mots sont ensuite "cachés" dans un ensemble de N séquences de longueur T, avec exactement une occurrence dans chaque séquence. Le challenge est alors, à partir des séquences uniquement, de retrouver les N mots insérés. Du point de vue de l'alignement local sans gap, il faut découvrir l'ULMA dont l'ensemble des occurrences correspond à l'ensemble des mots M_1, M_2, …, M_N. Les paramètres utilisés pour le challenge sont : $N = 20$, $T = 600$, $l = 15$ et $k = 4$ (appelé un signal-(20,600,15,4)). La performance d'une méthode est évaluée avec le coefficient de performance PC qui décrit le niveau de similarité entre l'ULMA caché et celui découvert. Plusieurs heuristiques spécifiques obtiennent de bons résultats sur ce problème. On peut citer parmi ces méthodes : PatternBranching (Price A. et al., 2003), Multiprofiler (Keich U. and Pevzner P.A., 2002), Projection (Buhler J. and Tompa M., 2002) et SMILE (Marsan L., 2002; Marsan L. and Sagot M.-F., 2000). Plusieurs articles (Buhler J. and Tompa M., 2002; Keich U. and Pevzner P.A., 2002; Pevzner P.A. and Sze S.-H., 2000; Price A. et al., 2003) annoncent de mauvaises performances pour des algorithmes non-spécifiques comme le Gibbs Site Sampler, MEME et CONSENSUS. Il faut toutefois prendre en compte plusieurs aspects lorsque l'on cherche à évaluer de tels algorithmes sur le problème challenge. Tout d'abord, les méthodes spécifiques et non-spécifiques ne sont pas équivalentes en terme de contraintes. Alors que les deux types d'approches utilisent les contraintes de la taille du motif et d'une occurrence par séquence, les approches spécifiques utilisent en plus le nombre de substitutions k dans les occurrences du motif caché. Lorsque cette contrainte n'est pas utilisée, le problème est beaucoup plus difficile à résoudre car le motif ne "ressort" plus de son environnement. Nous montrons clairement, dans (Hernandez D. et al., 2004), que pour MoDEL et le Gibbs Site Sampler, un signal-(20,600,15,4) est plus faible que d'autres ULMAs composés d'occurrences dans les parties aléatoires des séquences. En fait, le motif caché, avec ces paramètres et lorsqu'on n'utilise pas l'information du nombre de substitutions k, est au niveau du bruit vis-à-vis d'une mesure de contenu en information. Cela signifie que les méthodes non-spécifiques ne trouvent pas le motif caché parce que leur fonction objectif les guident vers des alignements de plus haute valeur. On peut donc en conclure que les comparaisons réalisées entre des méthodes spécifiques et des méthodes qui ne le sont pas sont fortement biaisées en faveur de ces premières.

Nous avons cependant montré que l'utilisation d'approches basées sur l'optimisation de fonction de contenu en information peut être très efficace pour découvrir un signal-(20,660,15,4). Nous avons développé une version modifiée de MoDEL qui est particulièrement efficace pour retrouver de tels motifs. Les modifications consistent uniquement à limiter l'exploration à l'espace Q et à utiliser la contrainte de k

substitutions mais uniquement comme condition d'arrêt et non pas pour guider l'exploration. Cela signifie que chaque mot exploré est testé pour vérifier s'il peut être apparié dans les *N* séquences avec *k* erreurs. En limitant le temps d'exploration maximum à deux minutes, MoDEL est capable de retrouver le motif caché dans 99.9% des cas (moyenne obtenue sur 1000 exécutions) avec un temps CPU moyen de 17 secondes (Pentium IV 1.5 Ghz).

3.2.3 TopMoDEL

3.2.3.1 Introduction

Les mécanismes de la biologie moléculaire sont en grande partie basés sur des interactions entre ou au sein des macro-molécules biologiques[27]. Ces interactions se font par fixation d'une molécule sur une ou plusieurs autres, déclenchant un mécanisme particulier ou formant de nouvelles molécules ou complexes de molécules. Elles sont aussi à l'origine des repliements intra-moléculaires déterminant la structure spatiale des molécules. Si ces mécanismes sont utiles à la survie de l'organisme, ils vont avoir tendance à être préservés par le processus de l'évolution. La fixation et le repliement des molécules sont réalisés par des interactions atomiques entre les molécules de base (acides aminés et nucléotide) des macro-molécules. Pour que ces interactions soit conservées, il faut, soit que les molécules concernées restent totalement inchangées dans les sites impliqués dans les interactions, soit que les changements dans ces molécules préservent les propriétés physico-chimiques nécessaires à l'interaction.

S'il existe des techniques expérimentales, comme la cristallographie, pour déterminer de façon certaine la structure tridimensionnelle d'une molécule ou les interactions entre molécules, ces techniques restent longues et coûteuses et ne peuvent donc pas être employées systématiquement. Le nombre important de séquences, aussi bien génomiques que protéiques, disponibles actuellement, rend nécessaire le développement de méthodes de prédiction automatiques des structures et des interactions, directement à partir des séquences primaires. Les règles d'appariement simples existant dans les séquences génomiques et plus particulièrement dans les ARNs ont permis la conception d'algorithmes de prédiction relativement fiables. Cela s'avère beaucoup plus difficile lorsqu'il s'agit de séquences protéiques. La plupart des méthodes de prédiction sur les protéines procèdent en fait par détection et/ou recherche de motifs de conservation locaux. Des modèles de motifs connus (comme les hélices α, les feuillets β, les HTH, les ponts disulfures,…), basés sur des représentations par réseau de neurones, SVM ou HMM, existent et sont utilisés pour faire des prédictions des sites correspondants. N'étant pas basées sur une représentation de l'interaction, ces méthodes sont limitées dans leur pouvoir de prédiction. Par exemple, si pour les feuillets β les modèles prédisent relativement bien la localisation des sites, ils sont incapables de déterminer comment les différents feuillets s'assemblent, c'est-à-dire quel feuillet s'apparie avec quel autre. Quelques approches ont été réalisées pour déterminer des corrélations au sein du même

[27] Elles peuvent également avoir lieu avec des molécules simples comme dans le cas de l'oxygène avec l'hémoglobine ou par l'intermédiaire de molécules simples comme dans le cas des ponts disulfure.

motif structurel (Afonnikov D.A. et al., 1997; Crooks G.E. and Brenner S.E., 2004; Korber B.T. et al., 1993; Larson S.M. et al., 2000). Elles se basent soit sur un alignement existant soit sur un motif déjà connu et localisé et utilisent des mesures basées sur les fréquences observées et attendues de paires de symboles ou sur l'information mutuelle. Il n'existe cependant pas, à notre connaissance, de méthode permettant de découvrir *de novo* automatiquement des motifs distants appariés dans les protéines.

Dans notre approche MoDEL nous cherchons à localiser des zones suffisamment conservées pour préserver les propriétés physico-chimiques. Nous voulons, avec la méthode TopMoDEL (Gras R. et al., 2003a), développée par Yoann Mescam durant sa thèse, étendre ce concept pour localiser des couples de sites conservant globalement leurs propriétés d'interactions par des mécanismes de mutations compensatoires. Ainsi, si une des molécules de base d'un des sites est mutée, soit cette mutation conserve les propriétés physico-chimiques nécessaires à l'interaction, soit une autre mutation intervient dans la position "miroir" de l'autre site qui rétablit la propriété d'interaction perdue. Nous souhaitons ainsi découvrir automatiquement des couples de sites appariés et donc participant à un mécanisme d'interaction moléculaire.

3.2.3.2 Découverte de multiples motifs

Les seules méthodes de découverte de motifs basées sur une mesure de contenu en information permettant la localisation de plusieurs motifs simultanément sont MEME et le Motif Sampler. Nous avons voulu profiter des avantages de MoDEL sur ces méthodes pour mettre au point un algorithme de découverte de multiples motifs. Le principe de cet algorithme est très semblable à celui de MoDEL à la différence près que nous utilisons ici la technique des niches écologiques par partage de fitness (Goldberg D.E. and Richardson J., 1987; Holland J.H., 1971) dans la partie algorithme génétique. Les niches écologiques dans les algorithmes évolutionnistes ont pour but de préserver de la diversité dans la population de solutions. Le partage de fitness consiste à diminuer la fitness des individus de la population en fonction du degré de ressemblance de chaque individu avec tous les autres. Cela ce fait par l'intermédiaire d'une fonction de partage *pa*. Un exemple d'une telle fonction entre deux individus X_1 et X_2 est donné ici :

$$pa(X_1, X_2) = 1 - \frac{d(X_1, X_2)}{n}$$

avec d(i_1,i_2) la distance de Hamming entre les deux vecteurs X_1 et X_2 et n la taille de ces vecteurs. La valeur de fitness d'un individu X transformée par partage de fitness est alors donnée par :

$$F(X) = \frac{F(X)}{\sum_{j=1}^{Tp} pa(X, Xj)}$$

avec *Tp* le nombre d'individus dans la population. Ainsi, plus un individu sera similaire à un grand nombre d'individus dans la population et plus sa fitness s'en trouvera diminuée. Il aura donc moins de chance d'être sélectionné pour la génération suivante.

Dans notre problème, la distance de Hamming n'est pas très révélatrice de la ressemblance entre deux individus de la population. En effet, dans MoDEL et TopMoDEL, un individu de la population est un mot de longueur fixée auquel on fait correspondre un alignement local sans gap dans l'ensemble des séquences. Ce que nous cherchons ce sont des alignements différents ayant un bon niveau de conservation. Or, les alignements peuvent être considérés comme différents s'ils se chevauchent pas ou peu (voir figure 30). Nous avons donc conçu une mesure de distance entre deux alignements basée sur le niveau de chevauchement entre eux. Elle est définie comme suit :

$$D(P_1, P_2) = \frac{\sum_{i=1}^{s} \frac{\max(0, W - |p_{1i} - p_{2i}|)}{\min(2 \cdot W, |p_{1i} - p_{2i} - W|)}}{s}$$

avec P_1 et P_2 deux ULMAs, p_{1i} et p_{2i} respectivement les débuts d'occurrences dans la ième séquence de P_1 et P_2, s le nombre de séquence dans l'alignement et W la longueur de l'alignement. Nous procédons ensuite comme pour le partage de fitness classique en remplaçant dans la fonction de partage la distance de Hamming par notre distance *D*.

TopMoDEL effectue donc la découverte de motif de la même façon que MoDEL, en ayant intégré le partage de fitness dans le calcul de la fitness des individus. TopMoDEL utilise un algorithme glouton pour déterminer quelles sont les solutions intéressantes dans la population. Il commence par sélectionner la solution de fitness maximale et par déterminer sa "niche", c'est-à-dire l'ensemble des autres solutions de la population qui lui sont très similaires, en utilisant la mesure de distance D. Puis il mémorise cette solution maximale dans la liste de résultats (si elle n'est pas déjà dedans) et il la retire, avec les autres solutions de sa niche, de la population. Le processus complet recommence à partir du nouveau maximum de la population restante jusqu'à ce que la population soit vide. TopMoDEL utilise également ces niches pour déterminer les couples de solutions auxquelles est appliqué l'opérateur de crossover. A partir d'un nombre fixé de générations, les crossovers n'ont plus lieu qu'entre solutions appartenant à la même niche. On garantit ainsi de n'utiliser le mixage entre solutions que lorsque les niches ont eu le temps de se former. On n'échange ainsi de l'information entre solutions que lorsque cela à un sens, c'est-à-dire lorsque les solutions représentent des alignements similaires. TopMoDEL fournit comme premier résultat la liste des motifs découverts selon ce principe, c'est-à-dire potentiellement tous les motifs de la longueur sélectionnée ayant un niveau de conservation supérieur au bruit et se chevauchant peu. Cette liste est ensuite utilisée dans une deuxième étape visant à découvrir les couples de motifs.

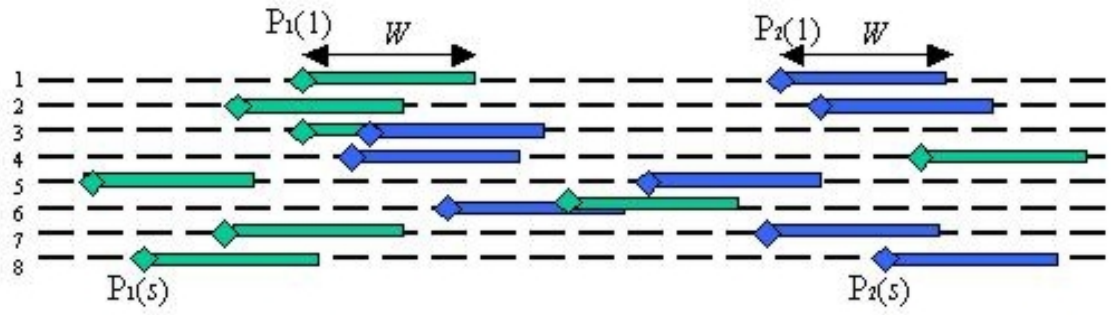


Figure 30. Exemple de deux alignements de longueur W dans un ensemble de 8 séquences et avec chevauchement dans les séquences 3 et 6.

3.2.3.3 Découverte de motifs dyadiques liés

Nous nous intéressons à la découverte de motifs dyadiques liés, c'est-à-dire des motifs composés de deux sites distants en interaction (Eskin E. and Pevzner P.A, 2002). Pour cela nous nous basons sur l'hypothèse des mutations compensatoires (Durbin R. et al., 1998; Woese C.R. and Pace N.R., 1993) et nous cherchons donc des couples d'alignements dont les compositions position à position co-varient. Le principe est de considérer deux à deux les colonnes des deux alignements, une dans chaque alignement et associées de manière directe ou palindromique (voir figure 31). Les couples d'alignements que nous recherchons sont alors ceux pour lesquels les couples de colonnes vont être en covariation, c'est-à-dire que les variations au sein d'une colonne vont correspondre aux variations dans l'autre colonne. Les symboles dans la première colonne ne doivent pas forcément être les mêmes que ceux de la deuxième mais c'est l'association entre les symboles qui doit être conservée. Par exemple le symbole 'a' de la première colonne aura toujours en vis-à-vis le symbole 'b' dans la deuxième colonne. Une manière de mesurer ce phénomène est de considérer les fréquences des couples de symboles dans les deux colonnes et de les comparer aux fréquences attendues si les colonnes n'étaient pas en covariation. La mesure communément utilisée provient de la théorie de l'information et s'appelle la mesure de l'*information mutuelle* (Chiu D.K.Y. and Kolodziejczak T., 1991; Gutell R.R. et al., 1992). L'information mutuelle $M(P_1(i),P_2(j))$ entre la colonne i de l'alignement P_1 et la colonne j de l'alignement P_2 est définie par :

$$M(P_1(i), P_2(j)) = \sum_{x_i, x_j} f_{x_i x_j} \cdot \log \frac{f_{x_i x_j}}{f_{x_i} \cdot f_{x_j}}$$

avec x_i,x_j les couples de symboles observés, f_{x_i} la fréquence de chaque symbole x_i dans la colonne i de l'alignement P_1, f_{x_j} la fréquence de chaque symbole x_j dans la colonne j de l'alignement P_2 et $f_{x_i x_j}$ la fréquence des couples de symboles. Si cette mesure s'avère très utile dans le cas où les alignements sont peu conservés, elle présente l'inconvénient

majeur de diminuer fortement lorsque les alignements sont très conservés. Pour pallier cet inconvénient, nous avons conçu une nouvelle mesure plus adaptée à mesurer la corrélation entre colonnes que nous recherchons. Notre mesure de corrélation $M_c(P_1(i),P_2(j))$ entre la colonne *i* de l'alignement P_1 et la colonne *j* de l'alignement P_2 est définie par :

$$M_c(P_1(i),P_2(j)) = \sum_{x_i,x_j} \log\left(\frac{f_{x_i x_j}}{2 \cdot f_{x_i}} + \frac{f_{x_i x_j}}{2 \cdot f_{x_j}}\right)$$

Nous mesurons finalement la corrélation entre deux alignements par la somme de cette mesure M_c pour chaque couple de colonnes associées.

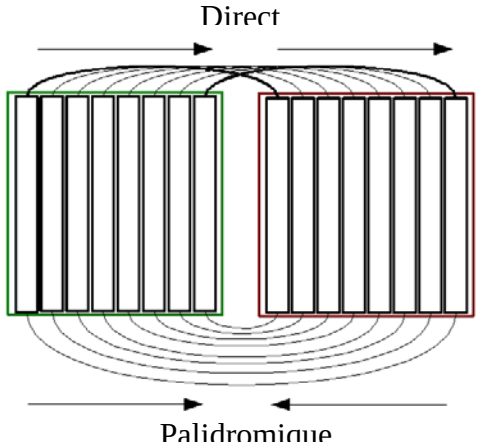


Figure 31. Association directe et palindromique des colonnes de deux alignements.

La deuxième étape de notre algorithme TopMoDEL consiste donc à mesurer, pour tous les couples d'alignements issus de la première étape, la somme de M_c pour tous les couples de colonnes associées selon le mode direct et le mode palindromique. Puis nous calculons la distribution de ces scores pour tous les couples formés. Nous calculons finalement le Z-score (nombre d'écarts types à la moyenne de cette distribution) de chaque couple d'alignement et donnons en résultat tous ceux qui sont au-dessus d'un seuil fixé.

3.2.3.4 Résultats

Pour évaluer les performances de notre méthode nous avons mis au point un système de génération de problèmes artificiels de difficultés paramétrables. Nous partons du principe que nous cherchons à découvrir les propriétés suivantes. Parmi l'ensemble des symboles possibles (l'ensemble des nucléotides ou des acides aminés selon que l'on considère des séquences génomiques ou protéiques), seul un sous-ensemble (non connu à l'avance) possédera les propriétés physico-chimiques nécessaires à une position donnée d'un site. Donc, dans une colonne donnée, seul un sous-ensemble des symboles doit apparaître et tous les symboles de ce sous-ensemble sont a priori interchangeables. Nous

appellerons ce sous-ensemble la *classe*. Il peut être différent d'une colonne à l'autre. Les symboles présents dans la colonne et n'appartenant pas à la classe sont considérés comme du bruit. Les colonnes que nous recherchons doivent donc majoritairement contenir des symboles appartenant à la classe et à chaque symbole de la classe de la colonne du premier alignement doit correspondre majoritairement un symbole de la classe de la colonne du deuxième alignement.

Notre générateur utilise deux paramètres. Le premier, %CC, correspond au pourcentage de symboles dans la colonne du premier alignement qui sont dans la classe et corrélés avec un symbole dans la classe de la colonne du deuxième alignement. Ce paramètre permet de doser le degré de conservation local de chaque site. Le deuxième, %HC, correspond au pourcentage de symboles dans les deux colonnes qui sont hors de la classe et donc sans corrélation. Ce paramètre permet de doser le degré de corrélation entre les deux sites du motif dyadique. En faisant varier ces deux paramètres nous pouvons étudier la capacité de notre algorithme à retrouver des alignements peu conservés et découvrir des couples d'alignements plus ou moins corrélés. La figure 32 présente un exemple de génération de deux colonnes.

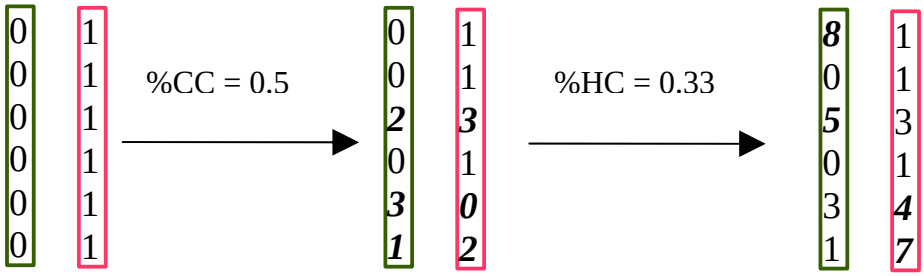


Figure 32. Exemple de génération de deux colonnes. Nous utilisons ici un alphabet de taille 9 : {0, 1, 2, 3, 4, 5, 6, 7, 8}. La classe est le sous-ensemble {0, 1, 2, 3} pour les deux colonnes. Nous commençons par générer deux colonnes complètement conservées (et donc complètement corrélées). Puis, nous changeons la valeur d'un pourcentage de %CC symbole dans la première colonne en changeant symétriquement la valeur correspondante dans la deuxième colonne de façon à ce que ces positions restent corrélées. Puis, dans les deux colonnes, un pourcentage de %HC symboles est changé dans un symbole aléatoire n'appartenant pas à la classe.

Nous donnons dans la figure 33 une vision de la valeur du score lorsqu'on fait varier les deux paramètres de 0 à 100. Dans cet exemple les deux colonnes sont de taille 100, l'alphabet est de taille 20 et la classe est de taille 4. Cela correspondrait donc à un alignement de 100 séquences protéiques dans lesquelles on considère qu'au plus 4 acides aminés peuvent porter une même propriété physico-chimique nécessaire à une position donnée d'un site d'interaction. Comme on peut le voir, notre score a bien les propriétés lui permettant de mesurer la corrélation indépendamment du niveau de conservation des sites. En effet, lorsque l'on fait varier le paramètre %CC, on change le niveau de conservation tout en préservant la corrélation. On remarque sur le graphique que la valeur du score reste stable lorsque %CC varie, ce qui est bien le comportement souhaité. De

même, lorsque %HC varie, c'est le niveau de corrélation qui varie. On remarque aussi que la valeur du score est maximale quand la conservation est maximale (%HC = 0) et décroît avec la valeur de %HC en étant minimale quand la conservation est minimale (%HC = 100). Ainsi, notre score est une mesure efficace du degré de corrélation tout en n'étant pas perturbée par le niveau de conservation locale des sites.

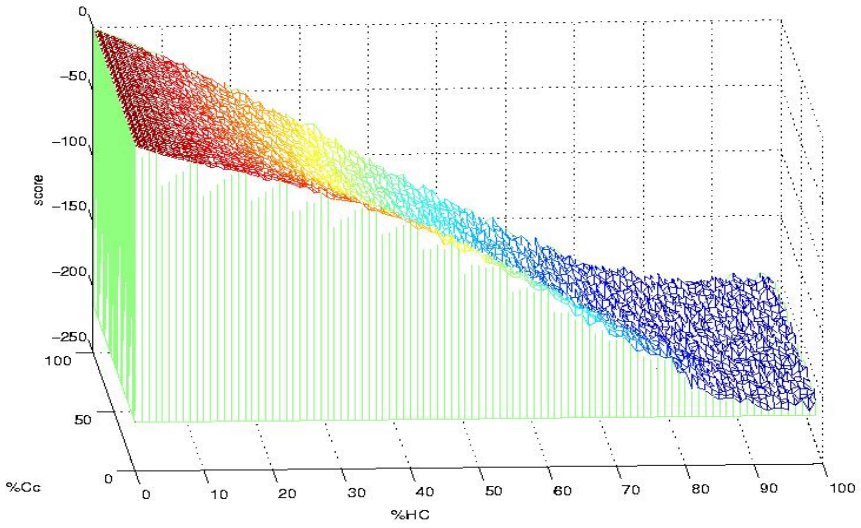


Figure 33. Valeur du score M_c lorsque l'on fait varier les paramètres %HC et %CC de 0 à 100. Cet exemple a été calculé sur deux colonnes de taille 100 utilisant un alphabet de taille 20 et une classe de taille 4.

Nous avons ensuite réalisé une expérience pour analyser la capacité de TopMoDEL à retrouver une série d'alignements plus ou moins bien conservés et, pour certains, corrélés avec un autre site. Dans cette expérience nous avons généré des ensembles de 20 séquences aléatoires de longueur 1000 utilisant l'alphabet des acides aminés. Dans chaque ensemble nous insérons 20 sites de longueur 15, représentant 10 motifs dyadiques liés, générés selon la méthode présentée dans la figure 31. Nous faisons varier les paramètres %CC et %HC de 0 à 100 et pour chaque combinaison de ces valeurs nous générons trois ensembles de séquences différents sur lesquels nous lançons une exécution de TopMoDEL de 50 générations. Les valeurs du graphe sont des moyennes pour ces trois exécutions.

La figure 34 présente les résultats en terme de couverture des sites insérés, c'est-à-dire que nous mesurons le pourcentage des positions des sites insérés qui sont couvertes par les sites découverts. On s'aperçoit que notre approche est capable de retrouver tous les sites insérés même avec un niveau de conservation relativement faible avec %CC = 30 et %HC = 50. Le taux de couverture baisse relativement vite lorsque %HC prend de plus grandes valeurs, ce qui correspond à un haut niveau de bruit. Le taux de couverture reste, par contre, de bonne qualité, même avec une valeur de %CC très élevée et lorsque que %HC est inférieur à 20 pourcents, avec entre 80 et 90 pourcents de couverture pour %CC

compris entre 60 et 100. Au-delà de ces valeurs le niveau de bruit devient extrêmement élevé et il devient impossible de découvrir les alignements.

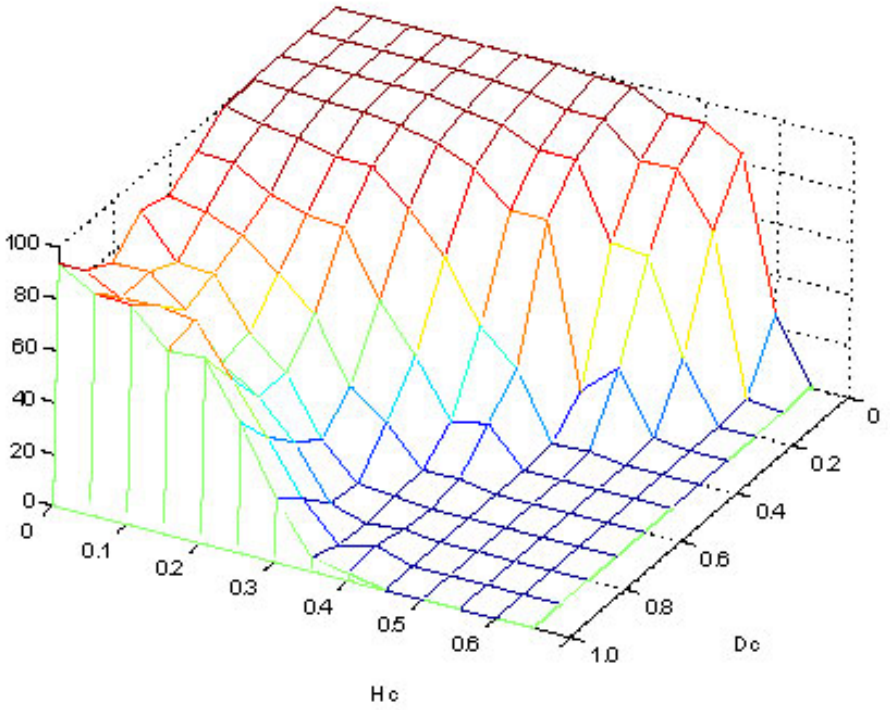


Figure 34. Pourcentage de couverture des sites insérés par les sites découverts lorsque l'on fait varier les valeurs de %CC et %HC (ici Dc et Hc) de 0 à 100. Chaque point du graphe est une moyenne sur trois exécutions de TopMoDEL pendant 50 générations sur un ensemble de 20 séquences d'acides aminés de longueur 1000 contenant 10 motifs dyadiques (20 sites) de longueur 15.

La figure 35 présente les résultats obtenus pour la découverte des motifs dyadiques liés au cours des mêmes tests que pour la figure 34. Dans ce cas, nous avons mesuré la valeur du score M_c pour tous les couples possibles à partir des alignements retrouvés. Puis nous avons calculé la valeur du Z-score obtenu par tous les couples correspondant effectivement aux motifs dyadiques liés insérés. Nous pouvons constater que ces Z-scores sont élevés (supérieurs à 2) lorsque le niveau de %HC reste faible (inférieur à 20) mais indépendamment du niveau de %CC. Nous sommes donc capables, non seulement de retrouver tous les alignements insérés même lorsque le niveau de bruit est relativement élevé, mais également, en utilisant notre score M_c, de prédire avec un bon niveau de fiabilité les alignements qui sont corrélés et cela quel que soit le niveau de variabilité dans la classe. Ainsi, si une grande majorité des séquences dans lesquelles on recherche les alignements, contiennent des sites en interaction, les positions concernées auront une variation éventuellement forte mais sur un petit nombre de symboles (ce qui correspond à une valeur de %CC élevée) et nous devrions donc être capables de les localiser. Une publication présentant ces travaux est actuellement en cours de rédaction.

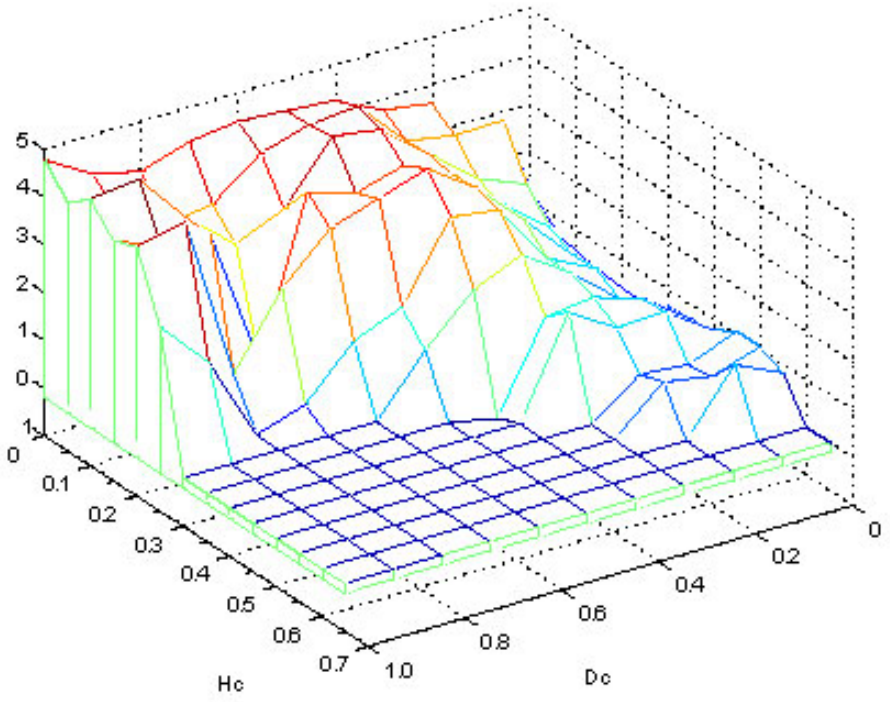


Figure 35. Z-score sur le score M_c obtenu pour les motifs dyadiques insérés lorsque l'on fait varier les valeurs de %CC et %HC (ici Dc et Hc) de 0 à 100. Chaque point du graphe est une moyenne sur trois exécutions de TopMoDEL pendant 50 générations sur un ensemble de 20 séquences d'acides aminés de longueur 1000 contenant 10 motifs dyadiques (20 sites) de longueur 15.

3.2.4 Conclusions

Nous avons développé deux méthodes, MoDEL et TopMoDEL, basées sur des techniques d'optimisation combinatoire stochastiques et utilisant des opérateurs dédiés, pour résoudre les problèmes de découverte d'alignements locaux sans gap et d'alignements locaux corrélés. Nous avons montré l'efficacité de ces approches pour résoudre ces problèmes et l'intérêt des algorithmes évolutifs lorsqu'ils sont couplés avec des opérateurs bien adaptés au problème à résoudre.

De nombreuses perspectives existent pour prolonger ces travaux. Nous travaillons actuellement à une nouvelle version de MoDEL permettant la levée des contraintes de la taille et du nombre d'occurrences par séquences. Nous réalisons, en cours d'exploration, une simulation de la distribution des scores d'information mutuelle pour des alignements de taille variable et ceux construits à partir d'un nombre d'occurrences variable. Associées avec de nouveaux opérateurs nous permettant d'explorer les dimensions de la longueur et du nombre d'occurrences, ces distributions vont nous permettre de découvrir l'alignement optimal sans avoir à spécifier sa longueur et son nombre d'occurrences. Nous avons également conçu un nouveau score d'information mutuelle dédié aux

séquences protéiques. Ce score permet de mesurer un niveau de conservation global d'un alignement en prenant en compte la similarité entre les acides aminés. Des premières expérimentations de MoDEL avec ce nouveau score donnent des résultats extrêmement prometteurs que ce soit par la signification biologique de l'alignement obtenu que par l'efficacité accrue de notre algorithme d'exploration. Deux publications sont actuellement en cours de rédaction pour présenter ces nouveaux résultats.

Nous devons également réaliser des comparaisons de TopMoDEL avec MEME et le Motif Sampler du point de vue de la capacité à découvrir de multiples alignements simultanément dans un ensemble de séquences. Nous allons aussi tester TopMoDEL sur des données réelles, que se soit pour essayer de découvrir des sites d'interactions intra-protéiques que pour des sites d'interactions inter-protéiques ou protéine-génomique. Nous inclurons les nouvelles fonctionnalités de MoDEL dans TopMoDEL dès qu'elles auront été validées.

CHAPITRE 4

Conclusions et perspectives

Les problèmes d'optimisation combinatoires sont et resteront des problèmes difficiles quelles que soient les méthodes algorithmiques mises au point pour les résoudre. Il apparaît en tout cas clairement qu'il est irréaliste d'imaginer un algorithme générique efficace pour tout type de problème de cette forme. Il existe cependant plusieurs pistes intéressantes pour définir une mesure de la complexité d'un problème donné. Cette mesure est basée sur différents concepts complémentaires comme le degré d'épistasie, la topologie des bassins d'attractions pour un opérateur de voisinage donné, la topologie du réseau d'interaction entre les variables du problème… Cette connaissance est particulièrement importante pour construire des algorithmes dédiés à chacun des problèmes. Cela passe par la création d'opérateurs d'exploration (au sens large, c'est-à-dire tout mécanisme permettant d'évaluer successivement des points de l'espace de recherche) prenant en compte ces connaissances pour réaliser une exploration adaptée à la structure du problème. Les algorithmes de type PMBGA ont été conçus pour essayer de découvrir automatiquement les opérateurs optimaux à partir d'une analyse de la structure de l'espace de recherche basée sur la construction d'un modèle probabiliste d'échantillons biaisés de l'espace de recherche. Cette approche, qui comme nous l'avons vu a des limitations importantes, offre toutefois des perspectives très intéressantes pour la résolution de problème comportant un nombre de dépendances raisonnables mais totalement inconnues. Nous avons, de plus, principalement testé cette approche dans un contexte de découverte du maximum global mais pas dans celui de la découverte de la meilleure solution possible dans un temps donné. Il serait donc également important d'analyser leurs performances dans ce contexte.

La bioinformatique est une source quasiment inépuisable de problèmes réels difficiles pouvant être abordés sous l'angle de l'optimisation combinatoire. Nous avons montré sur un certain nombre de problèmes bioinformatiques classiques l'apport indéniable que pouvaient avoir des techniques d'optimisation combinatoires stochastiques et métaheuristiques lorsqu'elles étaient conçues spécifiquement pour résoudre ces problèmes. Nous avons, malheureusement, pas encore eu le temps d'appliquer tous les concepts découlant de l'étude de la structure de l'espace de recherche à nos applications bioinformatiques. Cela permettrait, à la fois d'améliorer l'efficacité de nos méthodes de résolution mais serait également une aide pour la compréhension du problème biologique initial. En effet, le fait de mesurer la complexité d'un problème et de découvrir la structure de l'espace de recherche qu'il génère est en soit une information majeure sur la nature du problème. Les méthodes évolutives, par exemple, travaillant sur des échantillons de l'espace de recherche et en déduisant des propriétés de dépendance

entre les variables du problème, sont en mesure de fournir des connaissances nouvelles sur les mécanismes biologiques impliqués dans ces problèmes. La découverte de dépendances dans les variables du problème de la découverte de motif peut par exemple être à l'origine d'une classification des séquences concernées en fonction de leur contenu en motif mais également permettre la découverte simultanée de plusieurs motifs indépendants. Les dépendances découvertes lors de l'analyse de réseaux de régulation ou de la résolution des problèmes de "feature sélection" amèneraient des connaissances nouvelles sur les mécanismes d'interactions moléculaires. De façon générale, cette méthode peut être vue comme un outil de détection des corrélations entre les composants biologiques impliqués dans un processus quelconque.

De très nombreuses perspectives s'ouvrent à nous pour prolonger les travaux présentés dans ce mémoire que ce soit du point de vue des techniques d'optimisation combinatoire que de leur application pour la résolution de problèmes bioinformatiques. Nous souhaiterions travailler sur l'extension des concepts de découverte de dépendances aux approches de type programmation génétique qui ont un pouvoir d'expression bien plus élevé que ce que permettent les autres types d'algorithmes évolutifs. Des études plus complètes sur les capacités et les limitations des approches existantes de PMBGA seraient aussi importantes dans la perspective de la réalisation de nouveaux algorithmes plus performants. Plusieurs pistes sont possibles pour cela comme l'utilisation d'un modèle statistique basé sur une mesure plus informative que ne le sont les probabilités bayésiennes ou le développement d'un algorithme heuristique plus performant pour la construction du modèle. Il serait important de définir une classe de problèmes de référence, de difficulté paramétrable et prenant en considération les différents types de complexité existants, pour comparer équitablement les méthodes d'optimisation combinatoire. Cela pourrait être fait en s'inspirant du modèle de NK-landscape mais en y intégrant la notion de topologie des dépendances et un système de génération moins aléatoire, à même de générer des espaces de recherches structurés et donc explorables "intelligemment". L'intégration des notions de structure d'espace de recherche à nos méthodes bioinformatiques est également essentielle comme nous l'avons vu plus haut. Il serait tout aussi intéressant d'essayer nos approches sur de nouveaux problèmes bioinformatiques comme l'alignement multiple global, la prédiction de la structure des protéines, la découverte des réseaux de régulation, les problèmes de "feature selection"…

APPENDICE

Liste des publications

Articles dans des revues internationales

- Hernandez D., Gras R. **Neighborhood Functions and Hill-Climbing Strategies dedicated to the Generalized Ungapped Local Multiple Alignment,** European Journal of Operational Research, submitted.

- Yap Y.L., Wong M.P., Zhang X.W., Hernandez D., Gras R., Danchin A. **Conserved motifs in promoter regions of cancer markers in lung adenocarcinomas microarray – An ungapped local multiple alignment approach**, Nucleic Acid Research, accepted.

- Hernandez D., Gras R., Appel R.D. **MoDEL: An efficient strategy for ungapped local multiple alignment,** Computational Biology and Chemistry, 2004 vol 28(2), 119-128.

- Frey J., Gras R., Hernandez P., Appel R.D. **A Hierarchical Model of Parallel Genetic Programming Applied to Bioinformatic Problems**, LNCS 3019, Parallel Processing and Applied Mathematics, 1146-1153, 2004.

- Gras R., Hernandez D., Hernandez P., Zangger N., Mescam Y., Frey J., Martin O., Nicolas J., Appel R.D. **Cooperative metaheuristics for exploring proteomic data,** Artificial Intelligence Review 2003, vol 20 (1-2), 95-120.

- Hernandez P., Gras R., Frey J., Appel R.D. **Popitam: Towards new heuristic strategies to improve protein identification from tandem mass spectrometry data,** Proteomics 2003, 3 (6), 870-879.

- Müller M., Gras R., Binz P.-A, Hochstrasser D.F., Appel R.D. **Molecular Scanner Experiment with Human Plasma: Improving Protein Identification by Using intensity distributions of matching peptide masses,** Proteomics 2002,10, 1413-1425.

- Müller M., Gras R., Bienvenut W.V., Hochstrasser D.F., Appel R.D. **Visualization and Analysis of Molecular Scanner Peptide Mass Spectra**. J Am Soc Mass Spectrom 2002, vol 13, 221-231.

- Gras R., Guillet F., Gras R., Philippé J. **Réduction des colonnes d'un tableau de données par quasi-équivalence entre variables.** Extraction des connaissances et apprentissage 2001, 1(4), 197-202

- Gras R., Müller M. **Computational aspects of protein identification by mass spectrometry.** Current Opinion in Molecular Therapeutics 2001, 3(6), 526-532. (**invited author**)

- Gras, R., Mueller M., Gay S., Gasteiger E., Binz P.-A., Bienvenut W., Hoogland C., Sanchez, J.-C., Bairoch, A., Hochstrasser D.F., Appel R.D. **Improving protein identification from peptide mass fingerprinting through a parameterized multi-level scoring algorithm and optimized peak detection**. Electrophoresis 1999, 20, 3535-3550.

- Binz P.-A., Mueller M., Walther D., Bienvenut W.V., Gras R., Hoogland C., Bouchet G., Gasteiger E., Fabbretti R., Gay S., Palagi P., Wilkins M.R., Rouge V., Tonella L., Paesano S., Rossellat G., Karmime A., Bairoch A., Sanchez J.-C., Appel, R.D., Hochstrasser D.F. **A Molecular Scanner to Automate Proteomic Research and to Display Proteome Images**. Analytical Chemistry, Anal. Chem. 1999, Vol 71, 21, 4981-4988.

- Nicolas J., Delamarche C., Guerdoux-Jamet P., Gras R., **A symbolic-numeric approach to find patterns in genomes : Application to the translation initiation sites of E. coli** , Biochimie , Elsevier , Vol. 81 , 1999.

Chapitres de livres

- Gras R., Hernandez D., Hernandez P., Zangger N., Mescam Y., Frey J., Martin O., Nicolas J., Appel R.D. **Cooperative metaheuristics for exploring proteomic data,** Artificial Intelligence Methods and Tools for Systems Biology 2004, in press.

- Gras R., Hernandez P., Müller M., Appel R.D. **Scoring functions for mass spectrometric protein identification.** In: Handbook of Proteomics methods, Edited by P.M. Conn, Humana Press, 2003, pp 477-485.

- Déon C., Bienvenut W.V., Müller M., Gras R., Appel R.D., Sanchez J.-C., Hochstrasser D.F. **The Molecular Scanner: an automated high-throughput protein identification approach.** In: Protein analysis: a laboratory manual, Cold Spring Harbor Laboratory Press, in press.

- Bienvenut W.V., Müller M., Palagi P., Heller M., Gay S., Binz P.-A., Giron M., Gasteiger E., Jung E., Gras R., Sanchez J.-C., Appel R.D., Hochstrasser D.F. **Proteomics and mass spectrometry: some aspects and recent developments. In: Mass Spectrometry and Genomic Analysis**, N. Housby (Ed.), Kluwer Academic Publishers, The Netherland, 2001, pp. 1-53

Actes de conférences

- Hernandez P., Gras R., Appel R.D. **Popitam squints to better identify MS/MS spectra of modified peptides**, 5th Siena 2D electrophoresis meeting, Siena, Italy, 2004.

- Hernandez D., Gras R., Appel R.D. **Stratégies d'exploration de l'espace des alignements locaux multiples sans indels**, quatrièmes journées francophones de recherche opérationnelle, Fribourg, 2004, **invited speaker**.

- Gras R., Hernandez D., Hernandez P., Mescam Y., Appel R.D. **Cooperation in metaheuristics and application in bioinformatics,** European Conference on Combinatorial Optimization, 2004, **invited speaker**.

- Frey J., Gras R., Hernandez P., Appel R.D. **A Hierarchical Model of Coarse-Grained Parallel Genetic Programming.** 5th international conference on parallel processing and applied mathematics, Poland September 2003.

- Martin O., Gras R., Hernandez D., Appel R.D., **Optimizing Genetic Algorithms Using Self-Adaptation And Explored Space Modelization,** 5th international workshop on frontiers in evolutionary algorithms, North Carolina, September 2003, 291-294.

- Müller M, Gras R., Appel R.D. **Using peptide signal intensity distributions for better interpretation of molecular scanner data.** 5th Siena 2D electrophoresis meeting, Siena, Italy, 2002.

- Hernandez P., Gras R., Appel R.D. **Automated protein identification from tandem mass spectrometric data using bio-inspired algorithms.** 5th Siena 2D electrophoresis meeting, Siena, Italy, 2002.

- Gras R., Müller M., Hernandez D., Hernandez P., Zangger N., Palagi P., Binz P.-A., Lisacek F., Appel R.D. **New approaches of computational proteomics.** JOBIM 2002, St-Malo, 143-147.

- Müller M., Gras R., Bienvenut W.V., Hochstrasser D.F., Appel R.D. **Visualisation and Analysis of Molecular Scanner Peptide Mass Spectra.** Proc. of the Congress of the Swiss Proteomics Society, Geneva, 21-22 November 2001, 39-43.

- Gras R., Gasteiger E., Chopard B., Mueller M., Appel R.D. **New learning method to improving protein identification from peptide mass fingerprinting**. 4th Siena 2D electrophoresis meeting, Siena, Italy, Sept. 3 - 7, 2000.

- Bienvenut W.V., Heller M., Mueller M., Paesano S., Converset V., Gras R., Binz P.-A., Sanchez J.-C., Appel R.D., Hochstrasser D.F. **Comprehensive one step analysis by the molecular scanner: quantitative evaluation**. 4th Siena 2D electrophoresis meeting, Siena, Italy, Sept. 3 - 7, 2000.

- Gras R., Gras R., Gasteiger E., Binz P.-A., Mueller M., Chopard B., Appel R.D. **Classification automatique par algorithme génétique dans le cadre de l'identification par empreinte de masse peptidique. Journées Fouille dans les données par la méthode d'analyse statistique implicative**., pp 59-84, 2000, Caen, France.

- Gras R. **Multiple searches of association of flexible regular expressions in large genetic sequences**. MABS (Mathematical analysis of biological sequences) 1997, Rouen, France.

- Gras R. - **Recherche d'association de motifs approchés dans les grandes séquences génétiques**. RFIA'98, Volume III pp 355-364 Clermont-Ferrand.

- Gras R, Nicolas J. - **FOREST, a browser for huge DNA sequences**. The Seventh Workshop on Genome Informatics 96, pp 147-156 Tokyo.

- Gras R. - **Forest, FOuineur de REpétitions dans les Séquences Titanesques**. Journées Séminaire Junior en Intelligence Artificielle 96, Paris.

Abstracts

- Appel R.D., Binz P.A., Bouchet G., Catherinet S., Gras R., Hernandez C., Hernandez D., Hernandez P., Hoogland C., Lisacek F., Mostaguir K., Müller M., Palagi P., Pelhatre S., Tuloup M., Walther D., Zangger N. **Current developments at the Proteome Informatics Group,** Quelle informatique et bioinformatique pour la protéomique? Workshop, INRIA Rhône-Alpes, 2004.

- Hernandez P., Gras R., Appel R.D., **Popitam loves being fed on modified MS/MS spectra,** Peptide Fragmentation and Identified Workshop, Maryland 2004 accepted.

- Gras R., Frey J., Hernandez P., Appel R.D., **Un modèle hiérarchique "coarse-grained" de parallélisation d'algorithme de programmation génétique appliqué à des problèmes de bioinformatique.** Workshop: Résolution Parallèle des Problèmes NP-complets, Nice, October 2003.

- Gras R., Hernandez D., Hernandez P., Zangger N., Mescam Y., Frey J., Martin O., Nicolas J., Appel R.D. **Cooperative metaheuristics to solve complex bioinformatics problem.** International Summer School on Metaheuristics, Tenerife, Feb 2003.

- Appel R.D., Hernandez P., Tuloup M., Walther D., Gras R., Müller M., Hoogland C., Binz P.-A., Corthals G.L., Sanchez J.C., Hochstrasser D.F. **New developments in proteomatics at the Swiss Institute of Bioinformatics**. First Annual HUPO Congress, 21-24 November 2002, Versailles, France.

- Hernandez P., Gras R., Appel R.D. **Automated Protein Identification from Tandem Mass Spectrometric Data Using Bio-Inspired Algorithms.** Congress of the Swiss Proteomics Society 2002, Lausanne, 3-5 December 2002, p81-86.

- Zangger N., Gras R., Appel R.D. **Biological Sequence Clustering by Cooperative Agents.** Congress of the Swiss Proteomics Society 2002, Lausanne, 3-5 December 2002, p75-80.

- Zangger N., Gras R., Appel R.D. **A Symmetric Cooperative Metaheuristic for Biological Sequence Clustering Using Local Entropy as Similary Criterion.** NETTAB 2002, Bologna.

- Hernandez P., Gras R., Appel R.D. **Ant Colony Optimization Metaheuristic Applied to Automated Protein Identification from Tandem Mass Spectrometric Data.** NETTAB 2002, Bologna.

- Gras R., Müller M., Hernandez D., Hernandez P., Zangger N., Palagi P., Binz P.-A., Appel R.D. **Cooperative metaheuristic for proteomics.** Bioinformatics Workshop ISMIS 02, Lyon. (**invited speaker**)

- Hernandez D., Gras R., Lisacek F., Appel R. D. **MoDEL : Inférence de motifs avec un algorithme évolutionniste.** JOBIM 2002, St-Malo, 265-267.

- Müller M., Gras R., Appel R. D. **Vizualization and Interpretation of the Molecular Scanner Data.** ISMB2001 - The 9th International Conference on Intelligent Systems for Molecular Biology, Tivoli Gardens, Copenhagen, Denmark, July 22 - 25, 2001.

- Hernandez D., Gras R., Appel R. D. **Automated learning of unknown lenght motifs in unaligned biological sequences with genetic algorithm.** Proc. of the

Congress of the Swiss Proteomics Society, Geneva, 21-22 November 2001, 55-57.

- Hernandez D., Gras R., Appel R. **Automated learning of unknown length motifs in unaligned DNA sequences with genetic algorithms.** ISMB2001 - The 9th International Conference on Intelligent Systems for Molecular Biology, Tivoli Gardens, Copenhagen, Denmark, July 22 -25, 2001.

- Hochstrasser D.F., Bienvenut W., Mueller M. Gras R., Binz P.-A., Appel R.D., SanchezJ.C. **From Genome to Proteome: the development of a molecular scanner** Colloque AcM/MdA - Acquisition conduite par le Modèle - Model driven Acquisition, Grenoble, 23-24 novembre 2000.

- Palagi P.M., Hoogland Ch., Gras R., Mueller M., Binz P.-A., Fabbretti R., Bouchet G., Walther D., Gay S., Gasteiger E., Bairoch A., Sanchez J.-C., Hochstrasser D.F., Appel R.D. **Human proteomics and bioinformatics tools**. The first draft of the human genome : an academic and industrial perspective, Berlin, Germany, October 1 - 2, 2000.

- Binz P.A., Gasteiger E., Gras R., Hoogland C., Muller M., Walther D., Bienvenut W., Zimmermann C., Sanchez J.-C., Bairoch A., Appel R.D., Hochstrasser D.F. **Informatics of Proteomics and Possible Use in Laboratory Sciences. Computation in Clinical Laboratory** 2000 Third Millenium CCL, Milano, Sept. 21-23, 2000.

- Appel R.D., Walther D., Binz P.-A., Hoogland Ch., Gras R., Hochstrasser D.F. **Le rôle de la bioinformatique en protéomique**. Atelier de formation - Proteomics: new experimental approaches to unravel biological processes, Le Vésinet, 3-4 mai 2000.

- Appel R.D., Walther D., Binz P.-A., Hoogland Ch., Fabbretti R., Palagi P., Bouchet G., Gras R., Mueller M., Gay S., Gasteiger E., Bairoch A., Sanchez J.-C., Hochstrasser D.F. **High-throughput proteomics: the need for high-performance computing**. 27th SPEEDUP Workshop, Lugano, Switzerland, March 16 - 17, 2000.

- Gras R, Gasteiger E., Mueller M., R.D. Appel **New learning method to improve protein identification from peptide mass fingerprinting** The eighth International Conference on Computational Biology Intelligent Systems for Molecular Biology 00, La Jolla, California.

- Hochstrasser D.F., Binz P.-A., Mueller M., Walther D., Bienvenut W., Gras R., Hoogland Ch., Bouchet G., Gasteiger E., Fabbretti R., Gay S., Palagi P., Wilkins M., Rouge V., Tonella L., Paesano S., Rossellat G., Karmime A., Bairoch A., Sanchez J.-C., Appel R.D. **MALDI-TOF "Imaging": a Molecular Scanner for Proteomic Research** ABRF 2000.

- Appel R.D., Bairoch A., Walther D., Binz P.-A., Hoogland Ch., Fabbretti R., Palagi P., Bouchet G., Gras R., Mueller M., Gay S., Gasteiger E., Sanchez J.-C., Hochstrasser D.F. **Improving Proteomics Discovery with Bioinformatics**. Swiss Electrophores is Society (CHEL) meeting, Montreux Palace, Switzerland, Dec. 2nd 1999.

- Hochstrasser D.F., Sanchez J.-C., Binz P.-A., Bienvenut W., Gasteiger E., Müller M., Gras R., Fabretti R., Appel R.D. **The role of the virtual or computer "dry" laboratory in proteomics**. 31st Annual Meeting of USGEB 99.

- Binz P.A., Bienvenut W., Müller M., Gras R., Gasteiger E., Hoogland C., Fabbretti R., Karmime A., Paesano S., Rossellat G.,Rouge V., Tonella L., Bairoch A., Appel R., Sanchez J.C., Hochstrasser D.F. **Automation in Proteomics: the molecular scanner and the robotization of protein identification by peptide mass fingerprinting**. 3ème Journée d'Interaction des GRoupes de Recherche des Facultés de Médecine de Genève et Lausanne, Changins, Switzerland, October 7, 1999.

- Hochstrasser D.F., Binz P.-A., Bienvenut W., Mueller M., Walther D., Gras R., Hoogland Ch., Bouchet G., Gasteiger E., Fabbretti R., Gay S., Palagi P., Wilkins M., Rouge V., Tonella L., Paesano S., Rossellat G., Karmime A., Bairoch A., Appel R.D.,Sanchez J.-C., **Proteomics Technology Development in Geneva** Elektrophorese Forum'99, Munich, Germany, October 1999.

- Appel R.D., Bairoch A., Walther D., Binz P.-A., Hoogland Ch., Fabbretti R., Palagi P., Bouchet G., Gras R., Mueller M., Gay S., Gasteiger E., Sanchez J.-C., Hochstrasser D.F. **Improving, accelerating and automating proteomics discovery with bioinformatics** Electrophoreses Forum'99, Munich, Germany, October 1999.

- Gras R, Nicolas J. - **FOREST, browser of repeats in huge sequences**. The third International Conference on Computational Biology Intelligent Systems for Molecular Biology 95, Cambridge.

- Gras R, Nicolas J. – **Caractérisation de séquences biologiques. Définition d'un outil d'analyse pour les grandes séquences**. Rencontre Jeunes Chercheurs en Intelligence Artificielle 96, pp 263.

Rapports de recherche

- Gras R. - **Recherche multiple d'association d'expressions régulières flexibles dans les grandes séquences génétiques**. Rapport de recherche INRIA 1997, n^{o} 3240.

Invitation pour un séminaire

Université de Genéve, Université de Montreal, Université de Valenciennes, Université du Havre, Université de Lausanne, IRESTE Nantes, INRIA Rennes, INRIA Nancy, INRIA Grenoble, Institut Pasteur Paris, IDSIA Lugano, EPFL Lausanne.

Bibliographie

1. Ackley D.H. (1987). An empirical study of bit vector function optimization. In Genetic Algorithms and Simulated Annealing, Morgan Kaufmann), pp. 170-204.

2. Afonnikov D.A., Kondrakhin Y.V., Titov I.I., and Kolchanov N.A. Detecting direct correlation between positions in multiple alignment of amino-acid sequences. 87-98. 1997. Proceedings of the German Conference on Bioinformatics. Mewes H.W. and Frishman D.
   Ref Type: Conference Proceeding

3. Akutsu T., Arimura H., and Shimozono S. On approximation algorithms for local multiple alignment. 1-7. 2000. Proceeding 4th Int. Conf. Computational Molecular Biology.
   Ref Type: Conference Proceeding

4. Altenberg L. Fitness distance correlation analysis: an instructive counter-example. 57-64. 1997. Morgan Kaufmann. Proceedings of the 7th International Conference on Genetic Algorithms.
   Ref Type: Conference Proceeding

5. Altschul S.F., Gish W., Miller W., Myers E.W., and Lipman D.J. (1990). Basic local alignment search tool. J. Mol. Biol. *215*, 403-410.

6. Attwood T., Bradley P., Flower D., Gaulton A., Maudling N., Mitchell A., Moulton G., Nordle A., Paine K., Taylor P., Uddin A., and Zygouri C. (2003). PRINTS and its automatic supplement, prePRINTS. Nucleic Acids Res. *31*, 400-402.

7. Bafna V. and Edwards N. (2001). SCOPE: a probabilistic model for scoring tandem mass spectra against a peptide database. Bioinformatics *Suppl 1*, 13-21.

8. Bagley J.D. (1967). The behavior of adaptive systems which employ genetic and correlation algorithms. Dissertation Abstracts International *28*.

9. Bailey T.L. Likelihood vs. information content in aligning biopolymer sequences. cs93-318. 1993. University of California, San Diego.
   Ref Type: Report

10. Bailey T.L. and Elkan C. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. 28-36. 1994. AAAI Press. Proceedings of the Second

International Conference on Intelligent Systems for Molecular Biology.
Ref Type: Conference Proceeding

11. Bailey T.L. and Elkan C. (1995). Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization. Machine Learning Journal *21*, 51-83.

12. Baluja S. An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics. 1996. Proceedings of the Eleventh International Conference on Systems Engineering.
Ref Type: Conference Proceeding

13. Banzhaf W., Nordin P., Keller R.E., and Francone F.D. (1998). Genetic Programming. An Introduction. On the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann).

14. Barnes J.W., Laguna M., and Glover F. (1995). Intelligent Scheduling Systems. Brown D.E. and Scherer W.T., eds. Kluwer Academic Publishers), pp. 101-127.

15. Barnett L. Ruggedness and neutrality - the NKp family of fitness landscapes. 18-27. 1998. Proceedings of the sixth international conference on artificial life.
Ref Type: Conference Proceeding

16. Bartel C. (1990). Fast algorithm for peptide sequencing by mass spectrometry. Biomed. Environ. Mass. Spectrom. *19*, 363-368.

17. Bateman A., Birney E., Durbin R., Eddy S.R., Howe K.L., and Sonnhammer E.L. (2001). The Pfam Protein Families Database. Nucleic Acids Res. *28*, 263-266.

18. Battiti R. and Tecchiolli G. (1994). The reactive tabu search. ORSA Journal on Computing *6*, 126-140.

19. Bennington S.R. and Dunn M.J. (2001). Proteomics from Protein Sequence to Function. Springer-Verlag).

20. Bienvenut,W.V., Sanchez,J.C., Karmime,A., Rouge,V., Rose,K., Binz,P.A., and Hochstrasser,D.F. (1999). Toward a clinical molecular scanner for proteome research: parallel protein chemical processing before and during western blot. Anal. Chem. *71*, 4800-4807.

21. Binz,P.A., Muller,M., Walther,D., Bienvenut,W.V., Gras,R., Hoogland,C., Bouchet,G., Gasteiger,E., Fabbretti,R., Gay,S., Palagi,P., Wilkins,M.R., Rouge,V., Tonella,L., Paesano,S., Rossellat,G., Karmime,A., Bairoch,A., Sanchez,J.C., Appel,R.D., and Hochstrasser,D.F. (1999). A molecular scanner to automate proteomic research and to display proteome images. Anal. Chem. *71*, 4981-4988.

22. Blanchard J., Kuntz P., Guillet F., and Gras R. (2003). Implication intensity: from the basic statistical definition to the entropic version. In Statistical data Mining and Knowledge Discovery, Chapman and Hall, eds. (Washington: pp. 473-485.

23. Blickle T. and Thiele L. A Comparison of Selection Schemes used in Genetic Algorithms. TIK 11. 1995.
Ref Type: Report

24. Boeckmann B., Bairoch A., Apweiler R., Blatter M.C., Estreicher A., Gasteiger E., Martin M.J., Michoud K., O'Donovan C., Phan I., Pilbout S., and Schneider M. (2003). The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res. *31*, 365-370.

25. Bonabeau E., Dorigo M., and Theraulaz G. (1999). Swarm Intelligence. From Natural to Artificial Systems. Oxford University Press).

26. Brazma A., Jonassen I., Eidhammer I., and Gilbert D. (1998). Approaches to the automatic discovery of patterns in biosequences. J. Comput. Biol. *5*, 277-304.

27. Breci L.A., Tabb D.L., Yates J.R.III, and Wysocki V.H. (2003). Cleavage N-terminal to proline: analysis of a database of peptide tandem mass spectra. Anal. Chem. *75*, 1963-1971.

28. Bucher P. (1990). Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoters. J. Mol. Biol. *212*, 563-578.

29. Buhler J. and Tompa M. (2002). Finding Motifs Using Random Projection. J. Comput. Biol. *9*, 225-242.

30. Califano A. (2000). SPLASH: structural pattern localization analysis by sequential histograms. Bioinformatics *16*, 341-357.

31. Cantu-Paz E. and Goldberg D.E. On the Scalability of Parallel Genetic Algorithms. Evolutionary Computation. 1999.
Ref Type: Conference Proceeding

32. Cavicchio D.J. Adaptive search using simulated evolution. 1970. University of Michigan.
Ref Type: Thesis/Dissertation

33. Cheeseman P., Kanefsky B., and Taylor W. Where the really hard problems are. 331-337. 1991. Proceedings of the 12th IJCAI.
Ref Type: Conference Proceeding

34. Chen H., Gomes C., and Selman B. Formal Models of Heavy-Tailed Behavior in Combinatorial Search. 408-421. 2001. Cyprus. Proceedings of the International Conference on Principles and Practice of Constraint Programming.
Ref Type: Conference Proceeding

35. Chen T., Kao M.Y., Tepel M., Rush J., and Church G.M. (2001). A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. J. Comput. Biol. *8(3)*, 325-337.

36. Chickering D.M., Heckerman D., and Meek C. Learning Bayesian Network is NP-Hard. MSR-TR-97-07. 1997. Redmond, Microsoft Research .
Ref Type: Report

37. Chiu D.K.Y. and Kolodziejczak T. (1991). Inferring consensus structure from nucleic acid sequences. Computer Applications in the Biosciences *7*, 347-352.

38. Chvatal V. (1997). Resolution Search. Discrete Applied Mathematics *73*, 81-99.

39. Coello Coello C.A., Veldhuizen D.A.V., and Lamont G.B. (2002). Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publisher).

40. Colinge J., Masselot A., Giron M., Dessingy T., and Magnin J. (2003). OLAV: towards high-throughput tandem mass spectrometry data identification. Proteomics *3*, 1454-1463.

41. Colorni A., Dorigo M., and Maniezzo V. Distributed Optimization by Ant Colonies. 134-142. 1991. Proccedings of the 1991 European Conference on Artificial Life.
Ref Type: Conference Proceeding

42. Cook S.A. and Mitchell D.G. (1997). Finding Hard Instances of the Satisfiability Problem : A Survey. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.

43. Cotta C., Alba E., and Troya J.M. Stochastic Reverse Hillclimbing and Iterated Local Search. 2, 1558-1565. 1999. Proceedings of the IEEE Conference on Evolutionary Computing.
Ref Type: Conference Proceeding

44. Craig R. and Beavis R.C. (2003). A method for reducing the time required to match protein sequence with tandem mass spectra. Rapid Commun Mass Spectrom *17*, 2310-2316.

45. Craig R. and Beavis R.C. (2004). TANDEM: matching proteins with tandem mass spectra. Bioinformatics.

46. Creasy D.M. and Cottrell J.S. (2002). Error tolerant searching of uninterpreted tandem mass spectrometry data. Proteomics *2*, 1426-1434.

47. Crooks G.E. and Brenner S.E. (2004). Protein secondary structure: entropy, correlations and prediction. Bioinformatics *20*, 1603-1611.

48. Dancik V., Addona T., Clauser K., Vath J., and Pevzner P.A. (1999). De novo peptide sequencing via tandem mass spectrometry. J. Comput. Biol. *6*, 327-342.

49. Dantzig G.B. (1963). Linear Programming and Extensions. Princeton University Press).

50. Davidor Y. (1990). Epistasis variance: a viewpoint on representations, GA hardness, and deception. Complex Systems *4*, 369-383.

51. Davidor Y. Epistasis varience: a viewpoint on GA-hardness. 23-35. 1991. Morgan Kaufmann. Proceedings of the 1st Workshop on Foundations of Genetic Algorithms.
Ref Type: Conference Proceeding

52. Davis L. (1991). Handbook of Genetic Algorithm. (New York: Van Nostrand Reinhold).

53. De Jong E.D., Waston R.A., and Pollack J.B. Reducing Bloat and Promoting Diversity using Multi-Objective Methods. 2001. Genetic and Evolutionary Computation Conference. Morgan Kaufmann.
Ref Type: Conference Proceeding

54. De Jong K.A. An Analysis of the Behavior of a Class of Adaptive Systems. 1975. University of Michigan.
Ref Type: Thesis/Dissertation

55. Demassey S., Artigues C., and Michelon P. An Application of Resolution Search to the RCPSP. 2004. Beirut. The 17th European Conference on Combinatorial Optimization.
Ref Type: Conference Proceeding

56. Dorigo M., Di Caro G., and Gambardella L.M. (1999). Ant algorithms for discrete optimization. Artif. Life *5*, 137-172.

57. Durbin R., Eddy S., Krogh A., and Mitchison G. (1998). Biological sequence analysis.

58. Edman P. and Begg G. (1967). A Protein Sequenator. Eur. J. Biochem. *1*, 80-91.

59. Egelhofer,V., Gobom,J., Seitz H., Giavalisco, Lehrach H., and Nordhoff E. (2002). Protein identification by MALDI-TOF-MS peptide mapping: a new strategy. Anal. Chem. *74*, 1760-1771.

60. Eiben A.E. (2001). Evolutionary Algorithms and Constraint Satisfactions: Definitions, Survey, Methodology, and Research Directions. In Theoretical Aspects of Evolutionary Computating, pp. 13-30.

61. Elias J.E., Gibbons F.D., King O.D., Roth F.P., and Gygi S.P. (2004). Intensity-based protein identification by machin learning from a library of tandem mass spectra. Nat. Biotechnol. *22*, 214-219.

62. Eng J.K., McCormak A.L., and YatesIII J.R. (1994). An Approach to Correlate Tandem Mass Spectral Data of Peptides with Amino Acid Sequences in a Protein Database. J. American Society for Mass Spectrometry *5*, 976-989.

63. Eskin E. and Pevzner P.A (2002). Findings composite regulatory patterns in DNA sequences. Bioinformatics *18*, 354-363.

64. Fenyo D., Qin J., and Chait B.T. (1998). Protein identification using mass spectrometric information. Electrophoresis *19*, 998-1005.

65. Fenyo,D. (2000). Identifying the proteome: software tools. Curr. Opin. Biotechnol. *11*, 391-395.

66. Fernandez-de-Cossio J., Gonzalez J., and Besada V. (1995). A computer program to aid the sequencing of peptides in collision-activated decomposition experiments. CABIOS *11(14)*, 427-434.

67. Fernandez F., Tomassini M., Punch III W.F., and Sanchez J.M. Experimental Study of Multipopulation Parallel Genetic Programming. 283-293. 2000. Springer Verlag. Proceedings of the Third European Conference on Genetic Programming.
Ref Type: Conference Proceeding

68. Field H.I., Fenyo D., and Beavis R.C. (2002). RADARS, a bioinformatics solution that automates proteome mass spectral analysis, optimises protein identification, and archives data in a relational database. Proteomics *2*, 36-47.

69. Fogel G.B. and Corne D.W. (2003). Evolutionary Computation in Bioinformatics. (San Francisco: Morgan Kaufmann).

70. Fogel G.B., Weekes D.G., Varga G., Dow E.R., Harlow H.B., Onyia J.E., and Su C. (2004). Discovery of sequence motifs related to coexpression of genes using evolutionary computation. Nucleic Acids Res. *32*, 3826-3835.

71. Fogel L.J., Owens A.J., and Walsh M.J. (1966). Artificial Intelligence Through Simulated Evolution. (Chichester: Wiley).

72. Fonseca C.M. and Fleming P.J. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. Forrest S. 1993. San Mateo. Genetic Algorithms: Proceedings of the Fifth International Conference.
Ref Type: Conference Proceeding

73. Forrest S. and Mitchell M. (1993). Relative building-block fitness and building-block hypothesis. In Foundations of Genetic Algorithms 2, Morgan Kaufmann), pp. 109-126.

74. Frey J., Gras R., Hernandez P., and Appel R.D. (2004). A Hierarchical Model of Parallel Genetic Programming Applied to Bioinformatic Problems. Lecture notes in computer science *3019*, 1146-1153.

75. Frez B.J. (1998). Graphical Models for Machine Learning and Digital Communication. (Cambridge: MIT Press).

76. Gao Y. Threshold Phenomena in NK Landscapes. 2001. University of Alberta.
Ref Type: Thesis/Dissertation

77. Garey M.R. and Johnson D.S. (1979). Computers and Intractability: a Guide to the Theory of NP-Completeness. Freeman).

78. Gay S., Binz P.-A., Hochstrasser D.F., and Appel R.D. (1999). Modelling Peptide Mass Fingerprinting Data Using an Atomic Composition of Peptides. Electrophoresis *20*, 3527-3534.

79. Geard N., Wiles J., Hallinan J., Tonkes B., and Skellett B. A Comparison of Neutral Landscapes NK, NKp and NKq. 2002. Proceedings of the IEEE Congress on

Evolutionary Computation.
Ref Type: Conference Proceeding

80. Glover F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research *5*, 533-549.

81. Glover F. and Laguna M. (1997). Tabu Search. (London: Kluwer).

82. Glover F. and Tangedahl L. (1976). Dynamic Strategies for Branch and Bound. Omega *4*, 571-576.

83. Goldberg D.E. (1989a). Genetic Algorithm in Search, Optimization and Machine Learning., Addison-Wesley, ed.

84. Goldberg D.E. Sizing populations for serial and parallel genetic algorithms. 70-79. 1989b. Proceedings of the International conference on Genetic Algorithms.
Ref Type: Conference Proceeding

85. Goldberg D.E. (2002). The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers).

86. Goldberg D.E., Deb K., and Horn J. (1992). Massive Multimodality, Deception, and Genetic Algorithm. Parallel Problem Solving from Nature *2*, 37-46.

87. Goldberg D.E. and Richardson J. Genetic algorithms with sharing for multimodal function optimization. 41-49. 1987. Proceedings of the International Conference on Genetic Algorithms.
Ref Type: Conference Proceeding

88. Goldberg D.E., Sastry K., and Latoza T. On the supply of building blocks. 336-342. 2001. Proceedings of the Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

89. Goldberg D.E. and Segrest P. Finite Markov chain analysis of genetic algorithms. Grefenstette J.J. 1987. Proceedings of the Second International Conference on Genetic Algorithms.
Ref Type: Conference Proceeding

90. Golubski W. (2002). Genetic Programming: A Parallel Approach. Lecture notes in computer science *2311*.

91. Gras R. Aide à la détermination de la structure secondaire de macro-molécules biologiques par apprentissage automatique. 1994. Université de Rennes, France.
Ref Type: Report

92. Gras R. Un outil interactif de recherche de motifs dans les grandes séquences génétiques fondé sur l'arbre des suffixes. 1997. Université de Rennes, France.
Ref Type: Thesis/Dissertation

93. Gras R., Gasteiger E., Chopard B., Müller M., and Appel R.D. New learning method to improving protein identification from peptide mass fingerprinting. 2000. 4th Siena 2D

electrophoresis meeting.
Ref Type: Conference Proceeding

94. Gras R., Hernandez D., Hernandez P., Zangger N., Mescam Y., Frey J., Martin O., Nicolas J., and Appel R.D. (2003a). Cooperative metaheuristics for exploring proteomic data. Artificial Intelligence Review *20*, 95-120.

95. Gras R., Hernandez D., Hernandez P., Zangger N., Mescam Y., Frey J., Martin O., Nicolas J., and Appel R.D. (2004). Cooperative metaheuristics for exploring proteomic data. In Artificial Intelligence Methods and Tools for Systems Biology.

96. Gras R., Hernandez P., Muller M., and Appel R.D. (2003b). Scoring Functions for Mass Spectrometric Protein Identification. In Handbook of Proteomic Methods, Conn P.M., ed. Humana Press), pp. 477-485.

97. Gras R. and Muller M. (2001). Computational aspects of protein identification by mass spectrometry. Current Opinion in Molecular Therapeutics *3*, 526-532.

98. Gras R., Muller M., Gasteiger E., Gay S. , Binz P.A., Bienvenut W., Hoogland C., Sanchez J.C., Bairoch A., Hochstrasser D.F., and Appel R.D. (1999). Improving protein identification from peptide mass fingerprinting through a parameterized multi-level scoring algorithm and an optimized peak detection. Electrophoresis *20*, 3535-3550.

99. Greenberg H.J., Hart W.E., and Lancia G. (2004). Opportunities for Combinatorial Optimization in Computational Biology. INFORMS Journal on Computing *14*.

100. Gutell R.R., Power A., Hertz G.Z., Putz E.J., and Stormo G.D. (1992). Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods. Nucleic Acids Res. *20*, 5785-5795.

101. Hamm C.W., Wilson W.E., and Harvan D.J. (1986). Peptide sequencing program. CABIOS *2*.

102. Hanafi S. and Glover F. (2002). Resolution Search and Dynamic Branch-and-Bound. Journal of Combinatorial Optimization *6*, 401-423.

103. Harik G.R. Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. 1997. University of Michigan.
Ref Type: Thesis/Dissertation

104. Harik G.R., Cantu-Paz E., Goldberg D.E., and Miller B.L. The gambler's ruin problem, genetic algorithms, and the sizing of population. 7-12. 1997. Proceedings of the International Conference on Evolutionary Computation.
Ref Type: Conference Proceeding

105. Hernandez D., Gras R., and Appel R.D. (2004). MoDEL: An efficient strategy for ungapped local multiple alignment. Computational Biology and Chemistry *28*, 119-128.

106. Hernandez D., Gras R., Lisacek F., and Appel R.D. MoDEL: Inférence de motifs avec un algorithme évolutionniste. 265-267. 2002. JOBIM 2002.
Ref Type: Conference Proceeding

107. Hernandez P., Gras R., Frey J., and Appel R.D. (2003). Popitam: Towards new heuristic strategies to improve protein identification from tandem mass spectrometry data. Proteomics *3(6)*, 870-879.

108. Herrera F. and Lozano M. (1996). Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers. In Genetic Algorithms and Soft Computing, (New York: Physica-Verlag).

109. Hertz G.Z., Hartzell G.W., and Stormo G.D. (1990). Identification of consensus pattern in unaligned DNA sequences known to be functionally related. CABIOS *6*, 81-92.

110. Hertz G.Z. and Stormo G.D. (1999). Identifying DNA and Protein pattern with statistically significant alignment of multiple sequence. Bioinformatics *15*, 563-577.

111. Hines W.M., Falick A.M., Burlingame A.L., and Gibson B.W. (1991). Pattern-based algorithm for peptide sequencing from tandem mass spectra of peptides. J. American Society for Mass Spectrometry *3*, 326-336.

112. Hogg T., Huberman B.A., and Williams C.P. (1996). Phase transitions and the search problem. Artificial Intelligence *81*, 1-15.

113. Holland J.H. Hierarchical descriptions of universal spaces and adaptive systems. 1968. University of Michigan.
Ref Type: Report

114. Holland J.H. (1971). Processing and processors for schemata. In Associative information processing, Elsevier, ed., pp. 127-146.

115. Holland J.H. (1975). Adaptation in natural and artificial systems. The university of Michigan Press, Ann Arbor).

116. Huang L., Jacob R.J., Pegg S.C., Baldwin M.A., Wang C.C., Burlingame A.L., and Babbitt P.C. (2001). Functional assignment of the 20 S proteasome from Trypanosoma brucei using mass spectrometry and new bioinformatics approaches. J. Biol. Chem. *276*, 28327-28339.

117. Hughes J.D., Estep P.W., Tavazoie S., and Church G.M. (200). Computational identification of cis-regulatory elements associated with groups of functionally related genes in saccharomyces cerevisiae. J. Mol. Biol. *296*, 1205-1214.

118. Ishikawa K. and Niwa Y. (1986). Computer-aided peptide sequencing by fast atom bombardment mass spectrometry. Biomed. Environ. Mass Spectrom. *13*, 373-380.

119. James,P., Quadroni,M., Carafoli,E., and Gonnet,G. (1993). Protein identification by mass profile fingerprinting. Biochem. Biophys. Res. Commun. *195*, 58-64.

120. Jansen T. (2001). On Classifications of Fitness Functions. In Theoretical Aspects of Evolutionary Computing, Kallel L., Naudts B., and Rogers A., eds. Spriger), pp. 371-385.

121. Jensen F.V. (2001). Bayesian Networks and Decision Graphs. Springer).

122. Jensen L.J., Gupta R., Blom N., Devos D., Tamames J., Kesmir C., Nielsen H., Staerfeldt H.H., Rapacki K., Workman C., Andersen C.A.F., Knudsen S., Krogh A., Valencia A., and Brunak S. (2002). Predition of human protein function from post-translational modification and localization features. J. Mol. Biol. *319*, 1257-1265.

123. Jonassen I., Collins J.F., and Higgins D.G. (1995). Finding flexible patterns in unaligned protein sequences. Protein science *4*, 1587-1595.

124. Jones T. Evolutionary Algorithms, Fitness Landscapes and Search. 1995. University of New Mexico.
Ref Type: Thesis/Dissertation

125. Jones T. and Forrest S. Fitness distance correlation as a mesure of problem difficulty for genetic algorithms. 184-192. 1995. Morgan Kaufmann. Proceedings of the 6th International Conference on Genetic Algorithms.
Ref Type: Conference Proceeding

126. Jourdan L., Khabzaoui M., Dhaenens C., and Talbi E.-G. (2004). A hybrid metaheuristic for knowledge discovery in microarray. In Handbook of Bioinspired Algorithms and Applications, Olariu S. and Zomaya A.Y., eds. CRC Press USA).

127. Kallel L., Naudts B., and Reeves C.R. (2001). Properties of Fitness Functions and Search Landscape. In Theoretical Aspects of Evolutionary Computing, Kallel L., Naudts B., and Rogers A., eds. Springer), pp. 175-206.

128. Karlin S. and Altschul S.F. (1993). Applications and statistics for multiple high-scoring segments in molecular sequences. Proc Natl Acad Sci USA *90*, 5873-5877.

129. Kauffman S.A. (1989). Adaptation on rugged fitness landscapes. Lecture in Sciences of Complexity *1*, 527-618.

130. Kauffman S.A. (1993). The Origins of Order. In Self-Organization and selection in Evolution, (New York: Oxford University Press).

131. Keich U. and Pevzner P.A. Finding motifs in the twilight zone. 195-204. 2002. Proc. 6th Int. Conf. Computational Molecular Biology.
Ref Type: Conference Proceeding

132. Kimura S., Hatakeyama M., and Konagaya A. Inference of S-system Models of Genetic Networks using a Genetic Local Search . 631-638. 2003. Proceedings of 2003 Congress on Evolutionary Computation.
Ref Type: Conference Proceeding

133. Kimura S., Hatakeyama M., and Konagaya A. (2004). Inference of S-system Models of Genetic Networks from Noisy Time-series Data. Chem-Bio Informatics *4*, 1-14.

134. Kirkpatrick S., Gelatt C.D., and Vecchi M.P. (1983). Optimization by Simulated Annealing. Science *220*, 671-680.

135. Klose J. (2002). Protein Mapping by Combined Isoelectric Focussing and Electrophoresis in Mouse Tissues. A Novel Approach to Testing for Induced Point Mutations in Mammels. Humangenetik *26*, 231-243.

136. Korber B.T., Farber R.M., Wolpert D.H., and Lapedes A.S. (1993). Covariation of mutations in the V3 loop of human immunodeficiency virus type 1 envolope protein: an information theoretic analysis. Proc. Natl. Acad. Sci. U. S. A *90*, 7176-7180.

137. Koza J.R. (1992). Genetic Programming: on the programming of computers by means of natural selection. The MIT Press).

138. Kulback S. (1968). Information theory and statistics. (New York : Dover Publications).

139. Kuntz P., Gras R., and Blanchard L. Discovering Extend Rules with Implicative Hierarchies. 2002. Knoxville Tennesse. Conference on the new frontiers of statistical data mining and knowledge discovery.
Ref Type: Conference Proceeding

140. Kyte J. and Doolittle R.F. (1982). A simple method for displaying the hydropathic character of a protein. J. Mol. Biol. *157*, 105-132.

141. Langen H. Abstract book, 191. 1998. Sienna 2D electrophoresis meeting.
Ref Type: Conference Proceeding

142. Larson S.M., Di Nardo A.A., and Davidson A.R. (2000). Analysis of covariation in a SH3 domain sequence alignment: applications in tertiary contact prediction and the design of compensatory hydrophobic core substitutions. J. Mol. Biol. *303*, 433-446.

143. Lathrop R.H., Sazhin A., Sun Y., Steffen N., and Irani S.S. (2001). A Multi-Queue Branch-and-Bound Algorithm for Anytime Optimal Search with Biological Applications. Genome Informatics *12*, 73-82.

144. Lawrence C.E., Altschul S.F., Boguski M.S., Liu J.S., Neuwald A.F., and Wootton J.C. (1993). Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. Science *262*, 214.

145. Lawrence C.E. and Reilley A. (1990). An expectation Maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. Proteins *7*, 41-51.

146. Lin S.C., Punch III W.F., and Goodman D. Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach. Sixth IEEE parallel and distributed processing, 28-37. 1994.
Ref Type: Conference Proceeding

147. Lipman D.J. and Pearson W.R. (1985). Rapid and sensitive protein similarity searches. Science *227*, 1435-1441.

148. Mackey A.J., Haystead T.A., and Pearson W.R. (2002). Getting more from less: algorithms for rapid protein identification with multiple short peptide sequences. Mol. Cell Proteomics *1*, 139-147.

149. Magnin J., Masselot A., Menzel C., and Colinge J. (2004). OLAV-PMF: a novel scoring scheme for high-throughput peptide mass fingerprinting. J. Proteome Res. *3*, 55-60.

150. Manderick B. and Spiessens P. Fine-grained Parallel Genetic Algorithms. 428-433. 1989. Proceedings of theThird International Conference on Genetic Algorithms.
Ref Type: Conference Proceeding

151. Mann M, Hojrup P., and Roepstorff P. (1993). Use of mass spectrometric molecular weight information to identify proteins in sequence databases. Biol Mass spectrom *22*, 338-345.

152. Marsan L. Inférence de motifs structurés: algorithmes et outils appliqués à la détection de sites de fixation dans les séquences génomiques. 2002. Université de Marne-la-Vallée.
Ref Type: Thesis/Dissertation

153. Marsan L. and Sagot M.-F. Extracting structured motifs using a suffix tree - Algorithms and application to consensus identification. [Minoru S. and Shamir R.], 210-219. 2000. Tokyo, Japan, ACM Press. Proceedings of the Fourth Annual International Conference on Computational Molecular Biology (RECOMB 00).
Ref Type: Conference Proceeding

154. Martin O., Gras R., Hernandez D., and Appel R.D. Optimizing Genetic Algorithms Using Self-Adaptation And Explored Space Modelization. 291-294. 2003. North Carolina. 5th International Workshop on Frontiers in Evolutionary Algorithms.
Ref Type: Conference Proceeding

155. Matsuo T., Matsuda H., and Katakuse I. (1981). Computer program PAAS for the estimation of possible amino acid sequence of peptides. Biomed. Mass Spectrom. *8*, 137-143.

156. Mengshoel O.J. and Goldberg D.E. Probabilistic crowding: Deterministic crowding with probabilistic replacement. 409-416. 1999. Proceedings of the Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

157. Michalewicz Z. (1996). Genetic Algorithms + Data Structure = Evolution Programs. Springer).

158. Michalewicz Z. and Fogel D. (2000). How to Solve It: Modern Heuristics. Springer-Verlag).

159. Mitchell M., Forrest S., and Holland J.H. The royal road for genetic algorithms: Fitness landscapes and GA performance. Varela F.J. and Bourgine P. 245-254. 1992. MIT Press. Proceedings of the First European Conference on Artificial Life.
Ref Type: Conference Proceeding

160. Moey C.J. and Rowe J.E. (2004). Population aggregation based on fitness. Natural Computing *3* , 5-19.

161. Monasson R., Zecchina R., Kirkpatrick S., Selman M., and Troyansky L. (1999). Determining Computational Complexity From Characteristic Phase Transition . Nature *400*, 133-137.

162. Morgenstern B. (1999). DIALIGN 2: Improvement of the segment-to-segment approach to multiple sequence alignment. Bioinformatics. *15*, 211-218.

163. Muhlenbein H. How Genetic Algorithms Really Work: Mutation and Hill-Climbing. 15-25. 1992. Amsterdam. Parallel Problem Solving from Nature.
Ref Type: Conference Proceeding

164. Muhlenbein H. and Mahnig T. (2001). Evolutionary Algoritms: From Recombination to Search Distributions. In Theoretical Aspects of Evolutionary Computing, Kallel L., Naudts B., and Rogers A., eds. Springer), pp. 135-173.

165. Muhlenbein H., Mahnig T., and Rodriguez A.O. (1999). Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics *5*, 215-247.

166. Muhlenbein H. and Voigt H.M. (1996). Gene pool recombination in genetic algorithms. In Metaheuritics: Theory and Applications, Kelly J.P. and Osman I.H., eds. Kluwer Academic), pp. 53-62.

167. Mullan,L.J. (2002). Multiple sequence alignment--the gateway to further analysis. Brief. Bioinform. *3*, 303-305.

168. Muller M. Molecular Scanner Data Analysis. 2-5-2003. Swiss Institute of Bioinformatics, University of Geneva.
Ref Type: Thesis/Dissertation

169. Muller M., Gras R., Binz P.-A., Hochstrasser D.F., and Appel R.D. (2002). Molecular Scanner Experiment with Human Plasma: Improving Protein Identification by Using intensity distributions of matching peptide masses. Proteomics *2*, 1413-1425.

170. Muller,M., Gras,R., Bienvenut,W., Hochstrasser,D., and Appel,R.D. (2002). Visualization and Analysis of Molecular Scanner Peptide Mass Spectra. J. Am. Soc. Mass Spectrom *13*, 221-231.

171. Munetomo M., Murao N., and Akama K. (2004). Empirical Investigations on Parallelized Linkage Identification. Lecture notes in computer science *in press*.

172. Nelson R.W., McLean M.A., and Hutchens T.W. (2000). Biosensor chip mass spectrometry: a chip-based proteomics approach. Electrophoresis *21*, 1155-1163.

173. Neuwald A.F., Liu J.S., and Lawrence C.E. (1995). Gibbs motif sampling: detection of bacterial outer membrane protein repeats. Protein science *4*, 1618-1632.

174. Newman M. and Engelhardt R. (1998). Effect of neutral selection on the evolution of molecular species. Proc. R. Soc. London *256*, 1333-1338.

175. Notredame C. (2002). Recent progress in multiple sequence alignment: a survey. Pharmacogenomics *3*, 131-144.

176. Notredame C. and Higgins D.G. (1996). SAGA: Sequence Alignment by Genetic Algorithm. Nucleic Acids Res. *24*, 1515-1524.

177. Notredame C., Higgins D.G., and Heringa J. (2000). T-Coffee: A novel method for fast and accurate multiple sequence alignment. J. Mol. Biol. *302*, 205-217.

178. Notredame C., O'Brien E.A., and Higgins D.G. (1997). RAGA: RNA sequence alignement by genetic algorithm. Nucleic Acids Res. *25*, 4570-4580.

179. Pacheco P.S. (1997). Parallel Programming with MPI. (San Francisco: Morgan Kaufmann).

180. Palmer R. (1991). Optimization on rugged landscapes. In Molecular Evolution on Rugged Landscapes: Proteins, RNA and the Immune System, Perelson A.S. and Kauffman S.A., eds. Addison-Wesley), pp. 3-25.

181. Papadimitriou C.H. and Steiglitz K. (1998). Combinatorial Optimization: Algorithms and Complexity. Dover).

182. Pappin D.D.J., Hojrup P., and Bleasby A.J. (1993). Rapid identification of proteins by peptide-mass finger printing. Curr Biol *3*, 327-332.

183. Parker K.C. (2002). Scoring methods in MALDI peptide mass fingerprinting: ChemScore, and the ChemApplex program. J. Am. Soc. Mass Spectrom *13*, 22-39.

184. Pelikan M. Bayesian Optimization Algorithm: From Single Level to Hierarchy. 2002. University of Illinois.
Ref Type: Thesis/Dissertation

185. Pelikan M. and Goldberg D.E. Escaping hierarchical traps with competent genetic algorithms. 511-518. 2001. Proceedings of the Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

186. Pelikan M. and Goldberg D.E. Hierarchical BOA solves Ising spin glasses and MAXSAT. 2003. Proceedings of the Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

187. Pelikan M., Goldberg D.E., and Cantu-Paz E. BOA: The Bayesian Optimization Algorithm. I, 525-532. 1999. San Francisco, CA, Morgan Kaufmann. Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99.
Ref Type: Conference Proceeding

188. Pelikan M., Goldberg D.E., and Cantu-Paz E. (2000). Linkage Problem, Distribution Estimation, and Bayesian Networks . Evolutionary Computation *8*, 311-340.

189. Pelikan M. and Sastry K. Fitness Inheritance in the Baysian Optimization Algorithm. 48-59. 2004. Proceedings of the Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

190. Perkins D.N., Pappin D.D.J., Creasy D.M., and Cottrell J.S. (1999). Probability-based protein identification by searching sequence databases using mass spectrometry data. Electrophoresis *20*, 3551-3567.

191. Petricoin III E.F., Ardekani A.M., Hitt B.A., Levine P.J., Fusaro V.A., Steinberg S.M., Mills G.B., Simone C., Fishman D.A., Kohn E.C., and Liotta L.A. (2002). Use of Proteomic patterns in serum to identify ovarian cancer. Lancet *359*, 572-577.

192. Pevzner P.A., Dancik V., and Tang C.L. (2000). Mutation-tolerant protein identification by mass-spectrometry. J. Comput. Biol. *7*, 761-770.

193. Pevzner P.A. and Sze S.-H. Combinatorial approaches to finding subtle signals in DNA sequences. 269-278. 2000. San Diego. Proceedings of the eighth International Conference on Intelligent Systems for Molecular Biology.
Ref Type: Conference Proceeding

194. Press,W.H., Teukolsky,S.A., Vetterling,W.T., and Flannery,B.P. (1995). Numerical Recipies in C. (Cambridge: Cambridge University Press).

195. Price A., Ramabhadran S., and Pevzner P.A. (2003). Finding subtle motifs by branching from sample strings. Bioinformatics *19*, 149-155.

196. Prugel-Bennett A. and Rogers A. (2001). Modelling Genetic Algorithm Dynamics. In Theoretical Aspects of Evolutionary Computing, pp. 59-85.

197. Rechenberg I. (1973). Evolutionsstrategie: Optimierung Techiquer Systeme nach Prinzipien der Biologischen Evolution. (Stuttgart: Frommann-Holzboog verlag).

198. Reeves C. and Wright C. Epistasis in genetic algorithms: an experimental design perspective. Eshelman L.J. 217-230. 1995. 6th International Conference on Genetic Algorithms. Morgan Kaufmann.
Ref Type: Conference Proceeding

199. Rigoutsos I. and Floratos A. (1998). Combinatorial pattern discovery in biological sequences. Bioinformatics *14*, 55-67.

200. Rosinski J.A. and Atchley W.R. (1999). Molecular Evolution of Helix-Turn-Helix Proteins. J. Mol. Evol. *49*, 301-309.

201. Rowe J.E. (2001). The Dynamical Systems Model of the Simple Genetic Algorithm. In Theoretical Aspects of Evolutionary Computing, Springer), pp. 31-57.

202. Russel S. and Norvig P. (2003). Artificial Intelligence. A Modern Approach. Prentice Hall).

203. Sadygov R.G. and Yates J.R.III (2003). A hypergeometric probability model for protein identification and validation using tandem mass spectral data and protein sequence databases. Anal. Chem. *75*, 3792-3798.

204. Sakurai T., Matsuo T., Matsuda H., and Katakuse I. (1984). Paas 3: A computer program to determine probable sequence of peptides from mass spectrometric data. Biomed. Mass Spectrom. *11(8)*, 396-399.

205. Sastry K. and Goldberg D.E. Analysis of Mixing in Genetic Algorithms: A survey. 2002017. 2002. IlliGAL report.
Ref Type: Report

206. Sastry K. and Goldberg D.E. Designing Competent Mutation Operators via Probabilistic Model Building of Neighborhoods. 2004a. Proceedings of the Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

207. Sastry K. and Goldberg D.E. Let's Get Ready to Rumble: Crossover Versus Mutation Head to Head. 2004b. Proceedings of the Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

208. Sastry K., Goldberg D.E., and Pelikan M. Efficiency Enhancement of Probabilistic Model Building Genetic Algorithms. 2004. Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

209. Schwarz G. (197). Estimating the dimension of a model. The Annals of Statistics *6*, 461-464.

210. Searle B.C., Dasari S., Turner M., Reddy A.P., Choi D., Wilmarth P.A., McCormack A.L., David L.L., and Nagalla S.R. (2004). High-throughput identification of proteins and unanticipated sequence modifications using a mass-based alignement algorithm for MS/MS *de novo* sequencing results. Anal. Chem. *76*, 2220-2230.

211. Sharpe O.J. Towards a Rational Methodology for Using Evolutionary Search Algorithms. 2000. University of Sussex.
Ref Type: Thesis/Dissertation

212. Sherrington D. and Kirkpatrick S. (1975). Solvable model of a spin-glass. Phys. Rev. Lett. *35* , 1792-1796.

213. Shevchenko A., Sunyaev S., Loboda A.V., Shevchenko A., Bork P., Ens W., and Standing K.G. (2001). Charting the proteomes of organisms with unsequenced genomes by MALDI-quadrupole time-of-flight mass spectrometry and BLAST homology searching. Anal. Chem. *73*, 1917-1926.

214. Siegel M.M. and Bauman N. (1988). An efficient algorithm for sequencing peptides using fast atom bombardment mass spectral data. Biomed. Environ. Mass Spectrom. *15*, 333-343.

215. Sigrist C., Cerutti L., Hulo N., Gattiker A., Falquet L., Pagni M., Bairoch A., and Bucher P. (2002). PROSITE: a documented database using patterns and profiles as motif descriptors. Brief. Bioinform. *3*, 265-274.

216. Sinha S. and Tompa M. A Statistical Method for Finding Transcription Factor Binding Sites. 344-354. 2000. San Diego, AAAI Press. Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology.
Ref Type: Conference Proceeding

217. Soule T. and Foster J.A. (1999). Effects of Code Growth and Parsimony Pressure on Populations in Genetic Programming. Evolutionary Computation *6*, 293-309.

218. Stadler P. (1995). Towards a theory of landscapes. In Complex Systems and Binary Networks, Lopez-Pena R., Capovilla R., Garcia-Pelayo R., Waelbroeck H., and Zertouche F., eds., pp. 77-173.

219. Stracuzzi D.J. Some Methods for the Parallelization of Genetic Algorithms. http://www.cs.umass.edu/~stracudj/home_frame.html . 1998.
Ref Type: Electronic Citation

220. Sunyaev S., Liska A.J., Golod A., Shevchenko A., and Shevchenko A. (2003). MultiTag: multiple error-tolerant sequence tag search for the sequence-similarity identification of proteins by mass spectrometry. Anal. Chem. *75*, 1307-1315.

221. Tabb D.L., Saraf A., and Yates J.R.III (2003). GutenTag: High-Throughput Sequence Tagging via an Empirically Derived Fragmentation Model. Anal. Chem. *75*, 6415-6421.

222. Taylor J.A. and Johnson R.S. (1997). Sequence database searches via de novo peptide sequencing by tandem mass spectrometry. Rapid Commun Mass Spectrom *11*, 1067-1075.

223. Thierens D. (1999). Scalability Problems of Simple Genetic Algorithms. Evolutionary Computation *7*, 331-352.

224. Thierens D. and Goldberg D.E. Mixing in genetic algorithms. 38-45. 1993. Proceedings of the International Conference on Genetic Algorithms.
Ref Type: Conference Proceeding

225. Thierens D. and Goldberg D.E. Convergence models of genetic algorithm selection schemes. 116-121. 1994. Parallel Problem Solving from Nature.
Ref Type: Conference Proceeding

226. Thompson J.D., Higgins D.G., and Gibson T.J. (1994). CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. *22*, 4673-4680.

227. Thompson R.K. and Wright A.H. Additively decomposable fitness functions. 1996. University of Montana, Computer Science department.
Ref Type: Report

228. Tongchim S. and Chongstitvatana P. (2002). Parallel genetic programming: synchronous and asynchronous migration. J. of Artificial Life and Robotics *5*.

229. Tsutsui S., Yamamura M., and Higuchi T. Multi-parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms. Banzhaf W., Daida J., Eiben A.E., Garzon M.H., Honavar V., Jakiela M., and Smith R.E. 657-664. 1999. Morgan Kaufmann. Proceedings of the Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

230. Ukkonen E. (1993). Approximate string-matching over suffix-trees. Lecture notes in computer science 228-242.

231. Van Helden J., Andre B., and Collado-Vides J. (1998). Extracting regulatory sites from the upstream region of yeast genes by computational analysis of aligonucleotide frequencies. J. Mol. Biol. *281*, 827-842.

232. van Nimwegen E., Crutchfield J.P., and Mitchell M. (1997). Finite populations induce metastability in evolutionary search. Physics Letters A *229*, 144-150.

233. Vannechi L., Clergue M., Collard P., Tomassini M., and Verel S. Fitness Clouds and Problem Hardness in Genetic Programming. 690-701. 2004. LNCS 3103. Genetic and Evolutionary Computation Conference.
Ref Type: Conference Proceeding

234. Verel S., Collard P., and Clergue M. Where are bottlenecks in nk-fitness landscapes? 273-280. 2003. IEEE Press. IEE International Congress on Evolutionary Computation.
Ref Type: Conference Proceeding

235. Vose M.D. and Liepins G.E. (1991). Punctuated equilibria in genetic search. Complex Systems *5* , 31-44.

236. Walsh T. Search in a small world. 1172-1177. 1999. Morgan Kaufmann. Proceedings of the 16th International Joint Conference on Artificial Intelligence.
Ref Type: Conference Proceeding

237. Watts D.J. (1999). Small Worlds. The Dynamics of Networks between Order and Randomness. Princeton University Press).

238. Watts D.J. and Strogatz S.H. (1998). Collective dynamics of 'small-world' networks. Nature *393*, 440-442.

239. Weinberger E.D. NP completeness of Kauffman's n-k model, a tuneably rugged fitness landscape. SFI-TR-96-02-003. 1996. The Santa Fe Institute.
Ref Type: Report

240. Weiner P. Linear pattern matching algorithms. 1-11. 1973. 14th Annual symposium on switching and automata theory.
Ref Type: Conference Proceeding

241. Wilkins M.R., Gasteiger E., Bairoch A., Sanchez J.C., Williams K.L., Appel R.D., and Hochstrasser D.F. (1999). Protein identification and analysis tools in ExPASy server. Methods Mol Biol *112*, 531-552.

242. Wilkins M.R., Williams K.L., Appel R.D., and Hochstrasser D.F. (1997). Proteome Research: New Frontiers in Functional Genomics., Wilkins M.R., Williams K.L., Appel R.D., and Hochstrasser D.F., eds. Springer-Verlag).

243. Williams R., Gomes C., and Selman B. On the Connections Between Backdoors, Restarts and Heavy-Tailedness in Combinatorial Search. 2003. Italy. Proceedings of the 6th International Conference on Theory and Applications Of Satisfiability Testing.
Ref Type: Conference Proceeding

244. Woese C.R. and Pace N.R. (1993). Probing RNA structure, function, and history by comparative analysis. In The RNA World , Gesteland R.F. and Atkins J.F. , eds. Cold Spring Harbor Laboratory Press), pp. 91-117.

245. Wolpert D. and Macready W. (1997). No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 67-82.

246. Yagiura M. and Ibaraki T. (2001). On Metaheuristic Algorithms for Combinatorial Optimization Problems. Systems and Computers in Japan *32*, 33-55.

247. Yanev N. and Andonov R. Solving the Protein Threading Problem in Parallel. in press. 2003. Workshop on High Performance Computational Biology.
Ref Type: Conference Proceeding

248. Yates J.R., Griffin P.R., Hood L.E., and Zhou J.X. (1991). Computer aided interpretation of low energy ms/ms spectra of peptides. In Techniques in Protein Chemistry II, (San Diego: Academic Press).

249. Yates,J.R.I., Speicher,S., Griffin,P.R., and Hunkapiller,T. (1993). Peptide mass maps: a highly informative approach to protein identification. Anal. Biochem. *214*, 397-408.

250. Zhang,W. and Chait,B.T. (2000). ProFound: an expert system for protein identification using mass spectrometric peptide mapping information. Anal. Chem. *72*, 2482-2489.