Protein Structure Comparison: From Contact Map Overlap Maximisation to Distance-based Alignment Search Tool

Noël Malod-Dognin

Université de Rennes 1

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

Docteur de l'Université de Rennes 1

Spécialité : **Informatique**

préparée au laboratoire **INRIA Rennes - Bretagne Atlantique**

dans le cadre de l'École Doctorale **MATISSE**

présentée et soutenue publiquement
par

MALOD-DOGNIN Noël

le 29 Janvier 2010

Titre:
Protein Structure Comparison: From Contact Map Overlap Maximisation to Distance-based Alignment Search Tool.

Directeur de thèse: **Rumen ANDONOV**

Jury

Pr.	Dominique LAVENIER,	Président du jury
Dr.	Frédéric CAZALS,	Rapporteur
Dr.	Gunnar KLAU,	Rapporteur
Dr.	Jean-François GIBRAT,	Examinateur
Dr.	Miklós MOLNÁR,	Examinateur
Pr.	Rumen ANDONOV,	Directeur de thèse

Résumé étendu

La biologie structurale est riche en problèmes NP-complets qui sont habituellement résolus par des heuristiques dont l'efficacité est mise à mal par la taille des espaces de solutions. Afin de rendre applicables les approches proposées, notre objectif est de concevoir des algorithmes exacts et efficaces en se basant sur les techniques avancées de l'optimisation combinatoire.

En biologie structurale, il est couramment admit que la structure tridimensionnelle d'une protéine détermine sa fonction. Ce paradigme permet de supposer que deux protéines possédant des structures tridimensionnelles similaires peuvent partager un ancêtre commun et donc posséder des fonctions similaires. Estimer la similarité entre deux structures de protéines est donc une tâche primordiale pour laquelle la recherche a produit de nombreuses méthodes [27, 59, 60, 68], souvent très différentes les unes des autres, sans qu'aucune ne s'impose réellement. Parmi toutes les méthodes proposées, nous nous intéressons à la mesure de similarité appelée " maximisation du recouvrement de cartes de contacts " [28] (ou CMO), principalement parce qu'elle fournit des scores de similarité pouvant être utilisés pour obtenir de bonnes classifications automatiques des structures de protéines.

Dans cette thèse, la comparaison de deux structures de protéines est modélisée comme une recherche de sous-graphe dans des graphes k-partis spécifiques appelés graphes d'alignements. Ainsi modélisée, nous montrons que cette tâche peut être efficacement réalisée.

Le premier chapitre de cette thèse est dédié aux connaissances nécessaires sur les structures des protéines et leurs comparaisons. Nous proposons également un cadre général pour la comparaison des structures de protéines, basé sur les graphes d'alignements. Dans le second chapitre, nous modélisons CMO comme une recherche de sous-graphe maximum induit par les arêtes dans des graphes d'alignements, problème pour lequel nous proposons un solveur exact qui surpasse les autres algorithmes de la littérature. Néanmoins, la procédure d'alignement requière encore trop de temps de calculs pour envisager des comparaisons à grande échelle. Le troisième chapitre est consacré à l'accélération de CMO en utilisant des connaissances issues de la biologie structurale. Nous proposons notamment une approche hiérarchique basée sur les structures secondaires des protéines. Enfin, bien que CMO soit une très bonne mesure de similarité, les alignements qu'elle fournit possèdent souvent de fortes valeurs de déviation (root mean squared deviation, ou RMSD). Pour palier cette faiblesse, dans la dernière partie de cette thèse, nous proposons une nouvelle méthode de comparaison de structures de protéines basée sur les distances internes que nous appelons DAST (pour Distance-based Alignment Search Tool). Elle est modélisée en une recherche de clique maximum dans des graphes d'alignements pour laquelle nous présentons un solveur dédié montrant de très bonnes performances.

Chapitre 1 : Cadre général

1.1 Introduction

Estimer la similarité entre deux structures de protéines est une tâche primordiale pour laquelle la recherche a produit de nombreuses méthodes (souvent très différentes les unes des autres), sans qu'aucune ne s'impose réellement. Dans cette thèse, nous améliorons la mesure de similarité appelée " maximisation du recouvrement de cartes de contacts " (CMO). Pour cela, nous formalisons le concept de comparaison des structures de protéines et nous proposons un cadre général pour la comparaison des structures de protéines.

1.2 Principe général

Soient deux protéines P_1 et P_2 représentées par leurs ensembles ordonnés d'éléments (souvent leurs acides-aminés) V_1 et V_2. Une paire d'alignements $i \leftrightarrow k$ signifie que l'élément $i \in V_1$ est mis en correspondance (est aligné) avec l'élément $k \in V_2$. Un alignement est une séquence de paires d'alignements " $i_1 \leftrightarrow k_1, i_2 \leftrightarrow k_2, \ldots, i_n \leftrightarrow k_n$ " telle que pour tout $j < n-1$, $i_j < i_{j+1}$ et $k_j < k_{j+1}$. Le but de la comparaison de structures est d'obtenir à la fois un score et l'alignement correspondant. Le score mesure la similarité entre les deux protéines, et l'alignement représente ce qui est commun entre les deux protéines.

Un **graphe d'alignements** $N_1 \times N_2$ est un graphe $G = (V, E)$ dont l'ensemble des sommets V est décrit par une grille de N_1 lignes et N_2 colonnes, un sommet $i.k \in V$ se trouvant en ligne i et en colonne k. Entre deux sommets $i.k$ et $j.l$, une arête $(i.k, j.l)$ ne peut exister (c.a.d $(i.k, j.l) \in E$) que si $i < j$ et $k < l$.

La comparaison de structures protéiques se modélise dans un graphe d'alignements $|V_1| \times |V_2|$ de la manière suivante. Chaque ligne représente un élément de P_1, et chaque colonne représente un élément de P_2. Un sommet $i.k$ est dans V si et seulement si la paire d'alignements $i \leftrightarrow k$ est admissible (c.a.d. $i \in V_1$ est compatible avec $k \in V_2$). Une arête entre deux sommets $i.k$ et $j.l$ existe dans E si et seulement si : (1) $i < j$ et $k < l$, et (2) la paire d'alignements $i \leftrightarrow k$ est compatible avec la paire d'alignements $j \leftrightarrow l$. Il existe une multitude de méthodes d'alignements dans la littérature, et elles diffèrent principalement par:

1. La nature des éléments présent dans V_1 et V_2.

2. Les définitions de compatibilité entre éléments et de compatibilité entre paires d'alignements.

3. Le sous-ensemble de V correspondant à un alignement optimal entre P_1 et P_2.

Par exemple, rechercher des alignements tels que toutes les paires d'alignements soient mutuellement compatibles revient à chercher des cliques dans G, et l'alignement le plus long correspond à une clique maximal dans G.

1.3 Contributions

La formulation en graphe d'alignements pose un cadre général pour la comparaisons de structures de protéines. Ce cadre, basé sur des graphes k-partis particuliers, permet la conception d'algorithmes efficaces pour résoudre les problèmes de comparaisons de structures de protéines.

Chapitre 2 : La maximisation du recouvrement des cartes de contacts

2.1 Introduction

La maximisation du recouvrement de cartes de contacts (ou CMO, pour Contact Map Overlap maximisation) est une mesure similarité entre structures de protéines, proposée par Godzik dans [28]. C'est une mesure robuste car elle prend en compte les alignements partiels. Elle est invariante par translation et elle capture bien la notion intuitive de similarité. Quand une protéine se replie, des acides-aminés qui sont éloignés dans la séquence peuvent se retrouver très proches dans l'espace tridimensionnel. Ces relations de proximité sont capturées par une carte de contacts. La carte de contacts d'une protéine P est un graphe $G = (V, E)$, dont les sommets de V sont ordonnés et correspondent aux acides-aminés de P, et ou une arête (i, j) entre deux sommet i et j existe si et seulement si la distance euclidienne entre les acides-aminés i et j est inférieure à un certain seuil. Dans l'approche CMO, l'évaluation de la similarité entre deux protéines P_1 et P_2 se fait en déterminant le recouvrement maximum de leurs cartes de contacts $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$. La définition formelle est la suivante. Soit $I = (i_1, i_2, \ldots, i_s)$, $i_1 < i_2 < \ldots < i_s$, un sous ensemble de V_1, et $J = (j_1, j_2, \ldots, j_s)$, $j_1 < j_2 < \ldots < j_s$, un sous-ensemble de V_2. Dans l'alignement " $i_k \leftrightarrow j_k$ ", $\forall k \in [1; s]$, une arête (k, l) est commune (un recouvrement a lieu) si et seulement si les deux arêtes $(i_k, i_l) \in E_1$ et $(j_k, j_l) \in E_2$ existent. CMO consiste à trouver un alignement (et donc les sous ensembles I et J) dont le nombre d'arêtes communes soit maximum.

2.2 Problématique

CMO est un problème NP-difficile [29] et la création d'un solveur efficace qui garantisse la qualité de CMO est toujours un problème ouvert (un état de l'art est présenté dans [69]). Dans ce chapitre, nous nous focalisons sur la conception d'un algorithme exact pour résoudre CMO. Dans [14], Carr, Lancia et Istrail ont été les premiers à attaquer CMO par programmation linéaire en nombres entiers (PLNE). Cette approche semble prometteuse quand elle est couplée avec une méthode de relaxation lagrangienne [11, 12]. Une alternative a été récemment publiée dans [69], et propose une technique de réduction associée à un algorithme par séparation et évaluation.

Enfin, dans [63], CMO est reformulé comme une recherche de cliques maximum dans des graphes qui ressemblent à nos graphes d'alignements.

2.3 Principe général

CMO est modélisé dans un graphe d'alignements $|V_1| \times |V_2|$ de la manière suivante : Les élément de V_1 et de V_2 sont les acides-aminé des deux protéines P_1 et P_2. Toutes les paires d'alignements $i \leftrightarrow k$ sont autorisées (c.a.d : $\forall i \in V_1$ et $\forall k \in V_2$, $i.k \in V$). Une arête $(i.k, j.l)$ existe si et seulement si : (1) $i < j$ et $k < l$, et (2) $(i, j) \in E_1$ et $(k, l) \in E_2$. Résoudre CMO revient à chercher dans G un ensemble ordonné de sommets $\{i_1.k_1$, $i_2.k_2$, $\ldots$, $i_n.k_n\}$, nommé **ensemble de sommets croissants**, tel que (1) pour tout $j < n$, $i_j < i_{j+1}$ et $k_j < k_{j+1}$, et (2) le nombre d'arêtes correspondantes dans G soit maximal.

En se basant sur les propriétés des graphes d'alignements (par exemple, une solution admissible pour CMO ne peut contenir qu'au plus un sommet par ligne, et au plus un sommet par colonne), nous proposons un nouveau modèle de programmation linéaire en nombre entier. Il s'agit d'un problème du type $\max F(X, Y)$, ou les variables binaires $x_{ik} \in X$ représentent le fait de prendre ou non le sommet $i.k$ dans la solution, et les variables binaires $y_{ikjl} \in Y$ représentent le fait de prendre ou non l'arête $(i.k, j.l)$ dans la solutions. Lorsque ce problème est soumis à toutes les contraintes de CMO, le résoudre est NP-difficile. Nous proposons une relaxation lagrangienne (qui consiste à relâcher certaines contraintes tout en les introduisant dans la fonction objectif avec des coefficients λ) dont le problème relâché $\max F'(X, Y, \lambda)$ est résoluble en temps polynomial. Le problème original et le problème relâché sont liés par la propriété suivante : $\max F(X, Y) \leq \max F'(X, Y, \lambda)$, pour tout λ. Pour résoudre CMO, nous résolvons le problème dual associé qui est une minimisation sur λ de $\max F'(X, Y, \lambda)$ à l'aide d'une descente de sous-gradient. À chaque itération, étant donnée une valeur de λ, la solution optimale (en termes de X et de Y) de $\max(F'(X, Y, \lambda))$ est calculée, puis les contraintes relâchées et violées sont identifiées, et enfin les λ correspondants sont modifiés. Ce processus ce répète jusqu'à l'obtention de la solution optimale. Cependant, si la solution optimale de $\max(F(X, Y))$ n'est pas trouvée en moins de 4000 itérations de descente de sous-gradient, un processus par séparation et évaluation est mis en place. Ceci garantit de trouver la solution exacte, mais permet aussi de se comporter comme une métaheuristique en obtenant toujours une borne et une bonne solution.

2.4 Contributions

Dans un premier temps, nous modélisons CMO comme une recherche de sous-graphes induits par les arêtes dans un graphe d'alignements. Dans un second temps, en exploitant les caractéristiques des graphes d'alignements, nous proposons une nouvelle formulation de CMO à l'aide de la PLNE. Ensuite, nous présentons une relaxation lagrangienne efficace de notre modèle de PLNE qui est intégrée dans un algorithme par séparation et évaluation dédié. Sur des jeux de tests de la littérature, nous observons que notre algorithme est plus rapide que les quatre autres algorithmes exacts de la littérature précédemment cités [11, 14, 63, 69], et même plus rapide que certaines heuristiques récemment proposées [38, 54]. Les résultats obtenus montrent que notre solveur est à la fois plus rapide et propose également de meilleures bornes que [11]. Enfin, nous montrons que notre algorithme peut être utilisé en tant que heuristique pour déterminer rapidement des scores de similarité permettant d'obtenir des classifications automatiques en très bon accord avec la classification manuelle SCOP [49].

Les idées présentées dans ce chapitres ont été publiées dans:
R. Andonov, N. Yanev and N. Malod-Dognin, " An Efficient Lagrangian Relaxation for the Contact Map Overlap Problem ", K.A. Crandall and J. Lagergren (Eds.) : WABI 2008, LNBI 5251, p. 162-173, 2008.
Une version journal a été soumise à " Journal of computational Biology " et est en attente d'acceptation.

Chapitre 3 : Utilisation de connaissances biologiques

3.1 Problématique

Dans le chapitre précédant, nous avons montré que CMO est une bonne mesure de similarité, dans le sens ou les scores associés aux paires de structures de protéines permettent d'obtenir de bonnes classifications automatiques. Cependant, les comparaisons de paires de protéines sont encore trop coûteuses en temps de calcul pour permettre des comparaisons à grande échelle. Puisque CMO est un problème NP-difficile, une manière efficace pour accélérer le processus de résolution est de réduire la taille des données qui correspondent, dans notre cas, aux graphes d'alignements. Dans une première étape, nous nous inspirons de la biologie structurale en utilisant la structure secondaire des protéines pour définir deux filtres pour les sommets des graphes d'alignements. Dans une deuxième étape, afin de mieux exploiter l'information des structures secondaires, nous définissons une approche hiérarchique pour résoudre CMO. À chaque fois, le but est double: d'un côté le graphe d'alignements devient creux et donc le processus de résolution devient plus rapide, d'un autre côté, les alignements obtenus doivent être acceptables d'un point de vue biologique.

3.2 Principes des approches proposées

Le but du premier filtre est d'interdire les alignements entre les acides-aminés provenant d'une hélice-α et les acides-aminés provenant d'un brin-β. Étant donné une fonction $Type(P, i)$ qui associe à l'acide aminé i de la protéine P le type de structure secondaire à laquelle il appartient (hélice-α, brin-β ou boucle), cela revient à supprimer du graphe d'alignements des acides aminés les sommets $i.k$ tels que $Type(P_1, i) = \alpha$ et $Type(P_2, k) = \beta$, ou bien tels que $Type(P_1, i) = \beta$ et $Type(P_2, k) = \alpha$. Ce premier filtre autorise encore le mariage entre des acides-aminés provenant d'une structure secondaire et des acides aminés provenant d'une boucle. Le but du second filtre est d'interdire également de tels mariages. Cela est réalisé en supprimant du graphe d'alignements les sommets $i.k$ tels que $Type(P_1, i)$ $\neq Type(P_2, k)$.

L'approche hiérarchique, inspirée de celle de VAST [26], consiste à obtenir dans une première étape un alignement des éléments de structures secondaires (SSEs). Puis, dans une seconde étape, le graphe d'alignements des acides-aminés est filtré afin d'enlever les sommets qui contredisent l'alignement des SSEs. Comme il n'existe pas de méthodes d'alignement des SSEs basées sur les cartes de contacts dans la littérature, nous présentons une nouvelle méthode d'alignement des SSE basée sur des cartes de contacts afin de rester dans le contexte de CMO. Pour cela, la structure secondaire d'une protéine P est décrite par une carte de contacts valués qui est un graphe $G = (V, E, W)$ où les sommets de V sont ordonnés et représentent chacun une SSE de P. Deux SSEs i et j sont en contact (l'arête (i, j) existe dans E) si il existe au moins un contact entre leurs acides-aminés. Le poids $w_{ij} \in W$ de ce contact est égal au nombre total de contacts entre les acides-aminés de i et ceux de j. Étant donné deux cartes de contacts valués $G_1 = (V_1, E_1, W_1)$ et $G_2 = (V_2, E_2, W_2)$, à chaque contact commun entre deux arêtes $(i, j) \in E_1$ et $(k, l) \in E_2$ est associé un poids $w_{ikjl} = \min(w1_{ij}, w2_{kl})$. L'alignement de deux cartes de contacts valués est alors très similaire à celui proposé par CMO, la seule différence étant que l'on ne cherche plus à maximiser le nombre de contacts communs mais à maximiser la somme des poids sur les contacts communs. Ce problème, très similaire à CMO et accepté par notre modèle de PLNE, est résolu par une version modifiée de notre solveur A_purva.

3.3 Contributions

Nous proposons de réduire les graphes d'alignements en utilisant deux filtres basés sur les structures secondaires des protéines. Utiliser ces filtres permet d'accélérer jusqu'à 50 fois le processus de résolution de CMO. Les résultats obtenus nous ont incité à davantage exploiter les informations de structures secondaires et à proposer une approche hiérarchique pour CMO. Celle-ci consiste, dans une première étape, à aligner les éléments de structures secondaires (SSE), puis dans une deuxième étape, à utiliser cet alignement de structures secondaires pour filtrer le graphe d'alignements correspondant aux acides-aminés. Dans ce but, nous avons développé une nouvelle méthode d'alignement des structures secondaires basée sur des cartes de contacts valués. La maximisation du recouvrement de cartes de contacts valués est résolue par une version modifiée de l'algorithme proposé dans le deuxième chapitre. Nous montrons entre autres que les alignements de structures secondaires que nous obtenons permettent d'obtenir très rapidement des classifications automatiques des structures de protéines en très bon accord avec la classification SCOP [49].

Seul le premier filtre SSE a été présenté dans la conférence WABI 2008.

Chapitre 4 : Alignement basé sur les distances internes (DAST)

4.1 Problématique

Une des principales faiblesses de CMO est que dans le but de maximiser le nombre de contacts communs, elle peut introduire des “ erreurs ” comme aligner des acides-aminés qui sont proches dans l'espace tridimensionnel avec des acides-aminés qui ne le sont pas. Ces erreurs ont pour conséquence que les alignements générés ont souvent de grandes valeurs de RMSD (entre coordonnées ou entre distances internes). Ce problème de RMSD est notamment visible dans les résultats de comparaisons menées par Godzik dans [27], où les alignements générés par CMO sont généralement plus longs que ceux générés par d'autres méthodes, mais possèdent également des valeurs de RMSD (entre coordonnées) bien supérieures.

4.2 Principe

Les deux protéines P_1 et P_2 sont représentées par les ensembles ordonnés d'acides-aminés V_1 et V_2. Nous remplaçons la notion de contact commun, utilisée dans CMO, par celle de distance interne commune : soient deux acides-aminés i et j de P_1 et deux acides-aminés k et l de P_2. Si la distance euclidienne entre i et j (notée d_{ij}) est similaire à celle entre k et l (notée d_{kl}), plus ou moins un seuil τ, alors i et j partagent une distance interne commune avec k et l. Nous recherchons l'alignement le plus long tel que pour tout couple de paires d'alignements, les distances internes associées soient communes. La principale caractéristique de cette méthode d'alignement, que nous avons appelé DAST (pour Distance-based Alignment Search Tool), est que l'alignement obtenu possède obligatoirement un RMSD entre distances internes[1] inférieur ou égal à un seuil fixé τ.

DAST est alors modélisé dans un graphe d'alignements de la manière suivante : afin de réduire le taille du graphe d'alignements, deux acides-aminés $i \in V_1$ et $k \in V_2$ sont compatibles (c.-à-d. le sommet $i.k$ est dans V) si et seulement si i et k proviennent du même type de structure secondaire (hélice α, brin β ou boucle). Deux sommets $i.k$ et $j.l$ sont reliés par une arête $(i.k, j.l) \in E$ si et seulement si (1) $i < j$ et $k < l$, et (2) $|d_{ij} - d_{kl}| \leq \tau$. Rechercher l'alignement le plus long tel que pour tout couple de paires d'alignements, les distances internes associées soient communes revient à rechercher dans G une clique de cardinalité maximum.

Le problème de la clique de cardinalité maximum est l'un des premiers à avoir été prouvés NP-complet [40]. Dans un premier temps, nous exploitons les caractéristiques des graphes d'alignements pour proposer un nouveau modèle de programmation linéaire en nombres entiers pour résoudre les différents problèmes de cliques maximums (cardinalité, sommes des poids sur les noeuds et/ou sur les arêtes). Dans un deuxième temps, nous proposons un nouvel algorithme par séparation et évaluation que nous appelons ACF (pour Alignment Clique Finder) pour résoudre le problème de la clique de cardinalité maximum.

ACF peut être vu comme une généralisation en deux dimensions de l'algorithme d'Östergård [52], à laquelle nous avons ensuite ajouté des bornes profitant des spécificités des graphes d'alignements. Ces bornes se basent sur l'observation que dans un graphe d'alignements $N_1 \times N_2$ G = (V,E), la cardinalité de la clique maximum, dénotée par $|MCC(G)|$, peut être surestimée par les trois méthodes suivantes :

[1] $RMSD_d = \sqrt{\frac{1}{N_m} \times \sum(|d_{ij} - d_{kl}|^2)}$, où N_m est le nombre de couples de paires d'alignements "$i \leftrightarrow k, j \leftrightarrow l$".

1. $|MCC(G)| \leq \min(N_1, N_2)$, car il n'existe pas d'arêtes entre deux sommets provenant d'une même ligne ou d'une même colonne.

2. $|MCC(G)| \leq |LIS(G)|$, ou $|LIS(G)|$ dénote la cardinalité du plus grand ensemble de sommets croissants dans G.

3. $|MCC(G)| \leq |LIP(G)|$, ou $|LIP(G)|$ est la la cardinalité du plus grand ensemble de sommets croissants dans G, tel que deux sommets consécutifs soient toujours reliés par une arête.

La plus efficace de ces bornes, en termes d'accélération de la recherche de la clique maximum, est basée sur $|LIS(G)|$. Nous montrons également que la complexité en temps de cette borne est en $O(|V| \times \log(|V|))$.

4.3 Contributions

Nous introduisons une nouvelle méthode d'alignement des acides-aminés que nous appelons DAST. Étant donné un seuil τ, DAST cherche l'alignement le plus long tel que pour chaque couple de paires d'acides-aminés alignés ($i \leftrightarrow k$, $j \leftrightarrow l$), i et j partagent la même relation de distance que k et l (+/- τ). Par conséquent, le $RMSD$ (entre distances internes) de l'alignement est inférieur à τ. D'un point de vue algorithmique, DAST est modelé comme une recherche de cliques maximum dans un graphe d'alignements. En exploitant la structure particulière de ces graphes, nous présentons une nouvelle formulation en programmation linéaire en nombres entiers pour les problèmes de cliques maximums. Nous proposons également un nouvel algorithme de recherche de clique maximum dont les performances dépassent celles d'un des meilleurs algorithmes de recherche de clique maximum de la littérature.

Une première version du modèle de programmation linéaire en nombres entiers a été présentée dans la conférence " ROADEF 2008 " :
N. Malod-Dognin, R. Andonov, N. Yanev and J-F. Gibrat, " Modèle de PLNE pour la recherche de cliques de poids maximal ", ROADEF 2008, p.307-308, le 25 février 2008.
La version actuelle du modèle de programmation linéaire en nombres entiers, ainsi que les bornes utilisées dans l'algorithme par séparation et évaluation ont été présentées dans les conférences :
N. Malod-Dognin, R. Andonov and N. Yanev, " Maximum clique approach to protein structure similarity ", MASSEE (Mathematical Society of South-East Europe) International congress on mathematics, Ohrid, Macedonia, p.31, du 16 au 20 septembre 2009.
et
N. Malod-Dognin, R. Andonov and N. Yanev, " Maximum clique and similarity measures ", MMSC'09 International Workshop on Mathematical modelling and scientific computations, Velingrad, Bulgaria, p.22, du 23 au 26 septembre 2009.
Une version journal a également été acceptée :
N. Malod-Dognin, R. Andonov and N. Yanev, " Solving Maximum Clique Problem for Protein Structure Similarity ", Serdica Journal of Computing.
Enfin, la version finale de l'algorithme par séparation et évaluation a été acceptée

à la conférence “ SEA 2010 : the 9^{th} international Symposium on Experimental Algorithms ”.

Chapitre 5 : Discussions et travaux futurs

5.1 Discussions

5.1.1 Relations entre les différents modèles mathématiques pour les ensembles de sommets croissants

Dans CMO, il n’est pas nécessaire de modéliser des contraintes pour choisir une arête dont les extrémités participent dans l’ensemble de sommets croissants (il s’agit d’une conséquence de la maximisation de la fonction objectif). Il est à noter que ceci n’est pas toujours vrai. Par exemple, cette contrainte nécessite d’être explicitement modélisée pour les arêtes associées à des poids négatifs. Cela arrive notamment dans le modèle des alignements locaux développé par Guillaume Collet [16] pour la prédiction du repliement des protéines. Dans DAST, la définition d’une clique impose que deux sommets ne participent à l’ensemble de sommets croissants que s’ils sont connectés par une arête, ce qui nécessite l’ajout d’une contrainte de choix d’arête entre tout couple de sommets. Par conséquent, le modèle pour CMO est le plus général (c.-à-d. le moins contraint), suivit par le modèle des alignements locaux et, enfin, le modèle pour DAST est le plus spécifique (c.-à-d. le plus restrictif).

5.1.2 Solutions non-optimales

Notre solveur de CMO, A_purva, et notre solveur de clique, ACF, sont tous deux basés sur des algorithmes par séparation et évaluation. Cependant, ils diffèrent à la fois en termes de stratégie de séparations et en termes de nature des bornes utilisées pour les évaluations. Cela entraine une différence notable lorsque les processus de résolution sont arrêtés avant que la solution optimale ne soit trouvée. A_purva essaye dans un premier temps de trouver la solution optimale dans l’intégralité de l’espace de recherche (c.-à-d. dans la totalité du graphe d’alignements), en améliorant de façon itérative la borne inférieure (la meilleure solution admissible trouvée) et la borne supérieure (la plus petite solution relâchée trouvée). Si, lors de cette exploration, la solution optimale n’a pas été trouvée, alors l’espace de recherche est divisé et chaque sous-problème ainsi obtenu est résolut indépendamment. Si le processus de résolution est interrompu, alors A_purva peut retourner une solution admissible de bonne qualité qui correspond au graphe d’alignements dans son intégralité. Dans ACF, par contre, l’espace de recherche est tout d’abord divisé en sous-problèmes (le premier étant le sous graphe dont l’ensemble de sommet ne contient que le dernier sommet, le deuxième sous-problème étant le sous-graphe dont l’ensemble de sommet ne contient que les deux derniers sommets, etc.) qui sont ensuite résolus du premier au dernier. Cette fois-ci, si le processus de résolution est arrêtés, la solution retournée par ACF peut ne correspondre qu’à un sous-graphe très petit, et donc être très éloignée de la solution optimale dans la graphe d’alignements complet. Contrairement à A_purva, ACF ne peut donc pas être utilisé pour obtenir rapidement de bonnes solutions sub-optimales.

5.1.3 Utilisation des structures secondaires

Nous avons modélisé l'alignement des structures secondaires en tant qu'alignement stable (c.-à-d. qu'un élément ne peut être aligné qu'avec au plus un seul autre élément). Cependant, comme illustré dans le chapitre 3, la reconnaissance des structures secondaires n'est pas parfaite. Par exemple, une longue hélice-α peut être courbée, et les méthodes de reconnaissances de structures secondaires risquent alors de la considérer comme étant plusieurs petites hélices-α. Cela implique que la contrainte de stabilité dans l'alignement devrait être relâchée, à moins que la reconnaissance des structures secondaire ne s'améliore. Qui plus est, le début et la fin des éléments de structures secondaires sont encore mal détectés, et cela devrait être pris en compte, notamment au niveau des filtres basés sur les structures secondaires.

5.1.4 Préservation de l'ordre dans les alignements

Comme dans de nombreuses méthodes de comparaison de structures de protéines, et comme dans la définition originale de CMO, nous avons supposé que l'alignement entre deux structures de protéines doit conserver l'ordre entre les éléments alignés. En utilisant cette contrainte de préservation de l'ordre, nous avons pu créer des algorithmes efficaces pour la comparaison de structure de protéines. Cependant, relâcher cette contrainte pourrait permettre de détecter d'autres types de similarités entre protéines.

5.2 Travaux futurs

Nous sommes pleinement conscient que bien que nous ayons beaucoup travaillé l'aspect modélisation et l'aspect algorithmique de la comparaison de structures des protéines, il manque aux travaux présentés un point de vue biologie structurale permettant de confirmer la pertinence biologique des méthodes proposées.

Modéliser la comparaison des structures de protéines dans des graphes d'alignements nous a permis de proposer des algorithmes exacts très efficaces. Bien que notre solveur, A_purva, puisse être utilisé en tant que heuristique, cela n'a jamais été son objectif principal. Pour certaines applications, telle que la classification automatiques des structures de protéines à grande échelle, le temps d'exécution peut avoir une importance capitale. Nous pensons qu'il doit être possible, en exploitant les caractéristiques de nos graphes d'alignements, de proposer des heuristiques bien plus efficaces que celle disponibles actuellement.

Nous avons utilisé des informations issues des structures secondaires des protéines afin de réduire la taille des graphes d'alignements. Il serait intéressant d'utiliser d'autres types d'informations qui pourraient, par exemple, provenir des séquences des protéines (tels que les propriétés physicochimique des acides-aminés) ou bien des structures tridimensionnelles (tels que les angles dièdre entre acides-aminés).

Comme présenté dans les discussions, nous souhaitons relâcher la contrainte de préservation de l'ordre dans notre solveur de cliques maximum (ACF). Le but serait double : d'une part nous pourrions obtenir de nouveaux types d'alignements avec DAST et, d'autre part, ACF pourrait alors être appliqué sur des graphes plus généraux.

Enfin, puisque nous avons proposé un programme linéaire en nombres entiers pour le problème de la clique maximum, nous prévoyons la création d'un algorithme par séparation et évaluation avec relaxation lagrangienne pour le résoudre. Ce nouveau solveur pourrait ensuite être intégré dans DAST.

Contents

Chapter 1

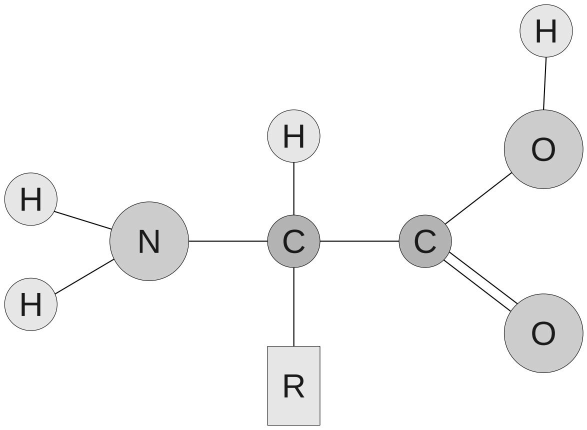
Introduction

In structural biology, it is commonly admitted that the three dimensional structure of a protein determines its function. A fruitful assumption based on this paradigm is that proteins sharing close three dimensional structures may derive from the same ancestor and thus, may share similar functions. Understanding function through structure is the primary goal of structural biology, and computing the similarity between two protein structures is one of the keys used for determining the homology relations between the proteins and for determining the function of a protein. Estimating the function of a protein is one of the keys among others for developing novel protein-based medical treatments. Consequently, determining the structural similarity between two proteins is a crucial task which has been extensively investigated by computational structural biologists. However, unlike the case of protein sequence comparisons, it is not yet clear what quantitative measure to use for comparing protein three dimensional structures, and a multitude of methods have been proposed, each aiming at capturing the intuitive notion of similarity.

In this thesis, and among all the proposed methods, we focus on the similarity measure called Contact Map Overlap maximisation (CMO) [28], mainly because it provides scores which can be used for obtaining good automatic classifications of the protein structures. In our approachs, comparing two protein structures is modelled as finding specific sub-graphs in specific k-partite graphs which we called alignment graphs, and we show that this task can be efficiently done by using advanced combinatorial optimisation techniques from operation research.

This introduction is divided in four parts. First, we introduce some structural biology knowledge about proteins which is necessary to undertand the rest of this study. Second, we both present the protein structure comparison problem and a short state of the arts of the proposed methods for solving it. Third, we propose a general canvas (based on alignments graphs) for modelling protein structure comparison problems, which is the basis of this thesis. Last, we introduce two advanced combinatorial optimisation techniques that we use, the integer programming approach and the Lagrangian relaxation. The second chapter of this thesis is then dedicated to the CMO problem which is modelled as a kind of maximum-edge-induced subgraph problem in an alignment graph. Based on this model, we propose both a novel integer programming formulation and an efficient Lagrangian relaxation-based solver which clearly outperforms previous algorithms from the literature, and which is a first step towards CMO based automatic classification of proteins. Even though we succeed to notably accelerate CMO, the procedure still

Figure 1.1: General structure of an α amino-acid.



The α carbon is the one connected to the R group. The amine group is on the left and the carboxyl acid group is on the right.

stays too much time consuming for large database comparisons. The third chapter of the thesis is dedicated to further accelerating CMO by using structural biology knowledge. Towards this goal we propose to reduce the alignment graph by using secondary structure knowledge and then we introduce a hierarchical approach for CMO which is also based on secondary structure of the proteins. Finally, although CMO is a very good scoring scheme, the alignments that it provides frequently possess large root mean square deviation values. To overcome this weakness, in the last chapter of the thesis, we propose a new comparison method based on internal distances which we call DAST (for Distance-based Alignment Search Tool). It is modelled as a maximum clique problem in alignment graphs, for which we design a dedicated solver with very good performances.

1 Protein composition, structures and functions

Since we are dealing with proteins, it is necessary to possess some basic knowledge about theirs compositions, structures and functions. Note that the notions presented here partly come from the book “Protein Structure and Function” [55].

1.1 Protein composition

An amino-acid is a molecule containing one amine group, one carboxylic acid group and one R group (sometime called sidechain). In an α amino-acid, the amine group and the carboxylic groups are attached to the same carbon atom, which is called the α-carbon (and denoted by C_α), as illustrated in **figure 1.1**. There are twenty different (in terms of R group) α amino-acids which enter into the composition of proteins. As illustrated in **table 1.1**, they are usually denoted by 1 letter codes. They are also classified according to common properties (big/small, polar/non-polar, etc...) as illustrated in the Venn diagram presented in **figure 1.2**, which was first

Table 1.1: The 20 α amino-acids found in proteins.

Name	3 letters code	1 letter code
alanine	ala	A
arginine	arg	R
asparagine	asn	N
aspartic acid	asp	D
cysteine	cys	C
glutamine	gln	Q
glutamic acid	glu	E
glycine	gly	G
histidine	his	H
isoleucine	ile	I
leucine	leu	L
lysine	lys	K
methionine	met	M
phenylalanine	phe	F
proline	pro	P
serine	ser	S
threonine	thr	T
tryptophan	trp	W
tyrosine	tyr	Y
valine	val	V

proposed in [65]. In the rest of this thesis, the term **amino-acid** will refers to the twenty α amino-acids which appear in proteins.

Proteins are polymers of amino-acids, linked together by peptide bonds in the order induced by the gene encoding for it. This means that a protein has a beginning, called *N* terminus, and an ending, called *C* terminus, and that the amino-acids of a protein can be labelled from first to last.

1.2 Protein structures

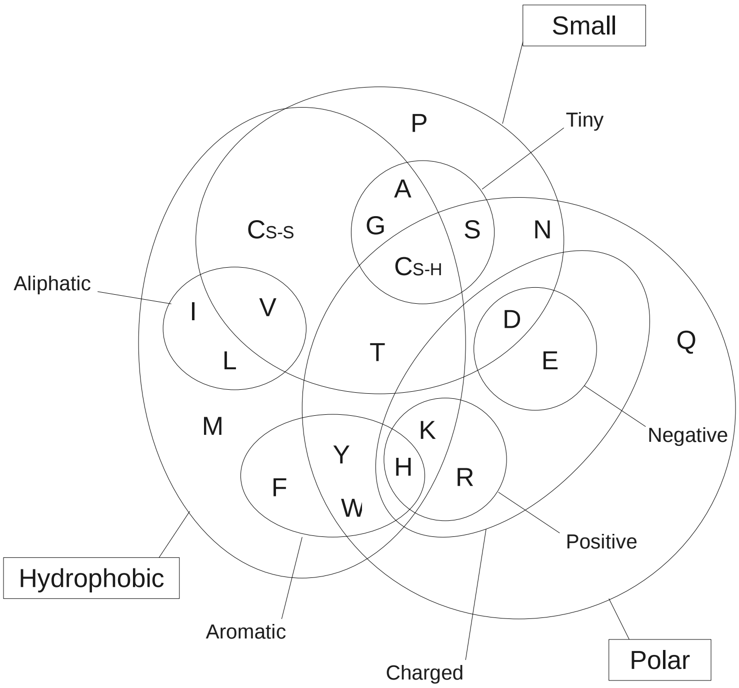
In the cells, at physiological temperatures and in aqueous solution, the polypeptide chains fold into specific 3D shapes which in most case are globular. The protein structure is divided into four levels, as illustrated in **figure 1.3**.

- The **primary structure** refers to the sequence of the different amino-acids in a protein chain, as determined by the sequence of nucleotides in the gene encoding it (see figure 1.3A).

- Thanks to regular strong hydrogen-bonding interactions betweens N-H and C=0 groups, parts of the protein chain adopt specific periodic motifs called α-helix and β-stands, also referred as secondary structure elements. The parts which are not in a secondary structure element are said to be loops. The **secondary structure** refers to the composition of a protein chain in terms of secondary structure element (see figure 1.3B).

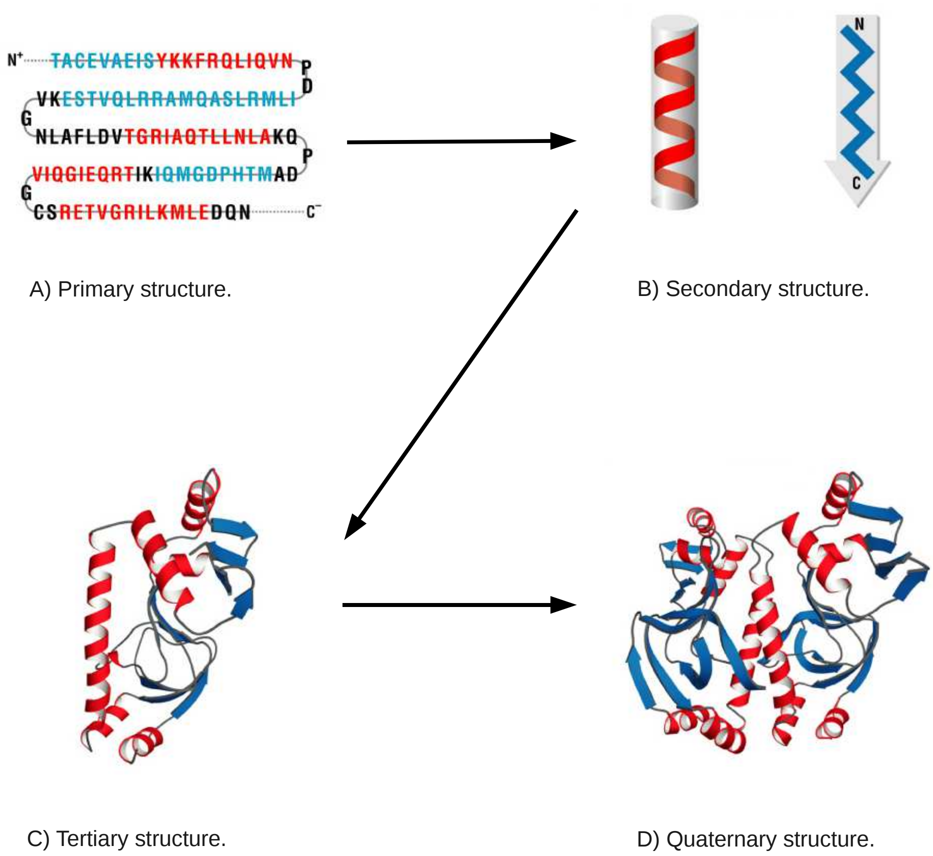
- The **tertiary structure** refers to the complete fold of a protein chain (see figure 1.3C). It represents how secondary structures and loops pack together, and is usually described by the 3D coordinates of the amino-acid atoms.

Figure 1.2: Venn diagram grouping amino-acids according to their properties.



The most important properties are small, hydrophobic and polar (and their counterparts). This Venn diagram grouping of amino acids is not the only one possible, but is probably the one that which most people would agree. Note that cystein is present in two positions (C_{S-S} and C_{S-H}). They represent the two oxidation states of cystein which have quite different properties.

Figure 1.3: The four levels of protein structure.



A) Primary structure.

B) Secondary structure.

C) Tertiary structure.

D) Quaternary structure.

Pictures taken from “Protein Structure and Function” [55]. **A)** The protein is represented by its ordered sequence of amino-acids, from it N terminus (or begining) to its C terminus (or ending). **B)** Hydrogen-bonding interactions define secondary structures element, like α-helices (in red) and β-strand (in blue). **C)** The fold of a protein chain alone. **D)** The fold of two or more protein chains together.

- The **quaternary structure** refers to the association of more than one folded protein chain into a single protein (see figure 1.3D), and is also described by the 3D coordinates of the amino-acid atoms.

Protein structures are frequently split into structural domains, which are units of protein structures able to fold stably as an independent entity in solution. These multidomain proteins probably appeared during the evolution by the fusion of genes that once coded for separate proteins. In this thesis, **protein structure** refers to the 3D coordinates of the amino-acids’ C_α from a single protein chain or from a single domain, whenever this coordinates comes from tertiary or quaternary structures.

1.3 Protein functions

Defining the function of a protein is not a simple task, partly because the function itself can be described at different level. From a biochemist point of view, the function is the biochemical role of an individual protein or of a protein complex. The four main biochemical role are :

1. **Binding** - The protein binds itself with another molecule called ligand. This is done by complementarity of shape and polar interaction, and thus, a binding protein only target a specific ligand.

2. **Catalysis** - The protein is used to change the speed of a chemical reaction, or even allow a reaction which should not be possible because of unfavourable thermodynamic conditions. Unlike other reagent that participates in the chemical reaction, a catalyst is not consumed by the reaction itself. Almost all chemical reactions in a living cell are catalysed.

3. **Information transfer** - Protein are flexible molecule, and their shape (also called conformation) may change in response to a change in pH or ligand binding. Such changes can be used to control cellular processes.

4. **Structural proteins** - Unlike other proteins which participate in the biochemical process of a cell, structural proteins are the building block of the cell itself. For example, actin and tubulin polymerise to form the cytoskeleton which allows the cell to maintain its shape and size.

From a geneticist point of view, function includes the biochemical role and the cellular role of a protein. Physiologist and developmental biologist may also include the phenotypic role of the protein (i.e. its effect on general properties of the organism).

1.4 Protein structure databases and classifications

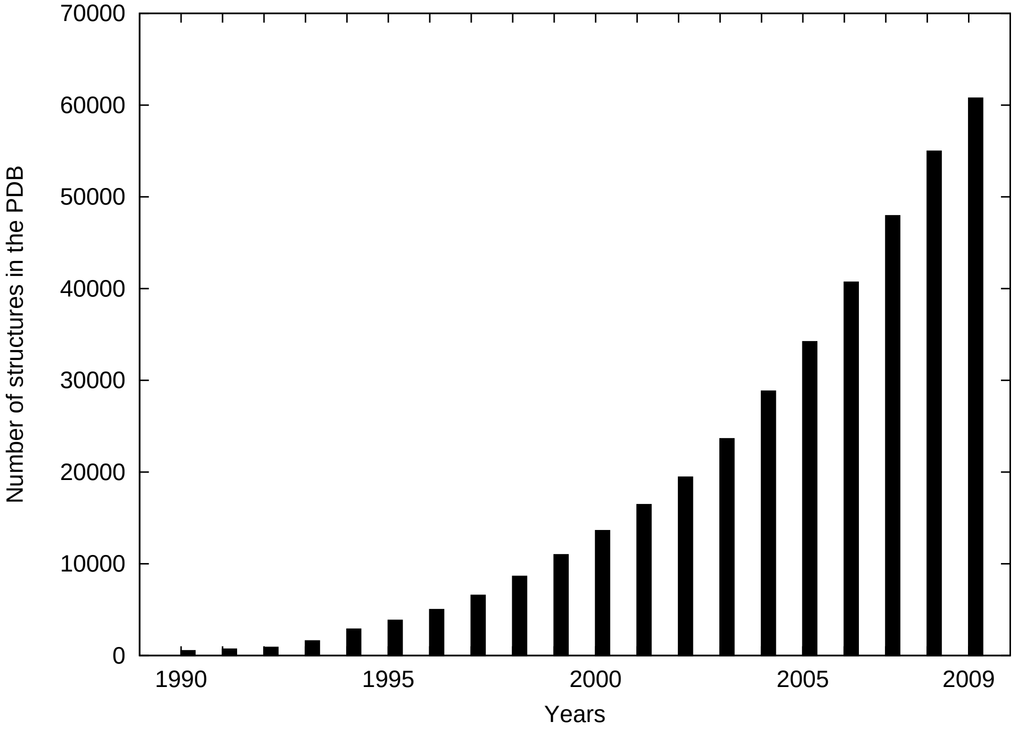
The main freely available protein structure database is the Research Collaboratory for Structural Bioinformatics Protein Data Bank (PDB)[1] [6]. Thanks to the recent progresses in molecular acquisition techniques like X-ray crystallography, nuclear magnetic resonance spectroscopy (NMR) and cryo-electron microscopy, the number of known structures has grown almost exponentially over the last 20 years, as illustrated in **figure 1.4**.

In order to function as a protein, a polypeptide must be able to form a stable tertiary structure under physiological conditions. Moreover, the demand of a protein function require that the folded protein should not be too rigid. Probably because of these opposite constraints, the number of folds adopted by proteins, though large, is very limited compared to all the possibilities of 3D shapes. This limited number of folds may results from physical constraints or from the limited divergent evolutions from pre-existing stable folds, and it is important in practice. First, if some stable folds are not represented in nature, it should be possible to create completely new proteins for industrial and medical applications. Second, it is then possible to classify proteins according to their structures.

Many protein structure classification have been proposed, but in this thesis, we will refer to the Structural Classification of Proteins (SCOP) [49] as a gold standard. It is a largely manual classification of protein structural domains based on similarities of their amino acid sequences (primary structure) and three-dimensional structures

[1] http://www.pdb.org

Figure 1.4: Data growth in the PDB during the last 20 years



During the last 20 years, the number of structure in the Protein Data Bank as grown rapidely. In october 13, 2009, there was 60756 structure in the PDB.

(tertiary or quaternary structure). Originally published in 1995, it is usually updated every two years by Alexei G. Murzin and his colleagues [4]. SCOP uses four levels of hierarchic structural classification:

1. **Class** - General “structural architecture” of the domain. Classes contain proteins having approximatively the same the composition in terms of secondary structure elements.

2. **Fold** - Major structural similarity. A fold contains proteins that have the same major secondary structures in the same arrangement and with the same topological connections. Different proteins with the same fold often have peripheral elements of secondary structure and turn regions that differ in size and conformation. In some cases, these differing peripheral regions may comprise half the structure. Proteins placed together in the same fold category may not have a common evolutionary origin: the structural similarities could arise just from the physics and chemistry of proteins favouring certain packing arrangements and chain topologies.

3. **Superfamily** - Probable common evolutionary origin. Proteins that have low sequence identities, but whose structural and functional features suggest that a common evolutionary origin is probable are placed together in superfamilies. For example, actin, the ATPase domain of the heat shock protein, and hexakinase together form a superfamily.

4. **Family** - Clear evolutionarily relationship. Proteins clustered together into families are clearly evolutionarily related. Generally, this means that pairwise residue identities between the proteins are 30% and greater. However, in some cases similar functions and structures provide definitive evidence of common descent in the absence of high sequence identity; for example, many globins form a family though some members have sequence identities of only 15%.

There are now a number of other databases which classify protein structures, such as CATH [51], FSSP [36] and DDBASE [62], however, the distinction between evolutionary relationships and those that arise from the physics and chemistry of proteins is a feature that is only available in SCOP and in CATH. Human expertise is still needed to decide whether certain proteins are evolutionary related and therefore should be assigned to the same superfamily, or their similarity is a result of structural constraints and therefore they belong to the same fold. This human expertise explains why only 63 percents of the PDB structures (as illustrated in **table 1.2**) are classified in SCOP.

2 Protein structure comparison

The experiments required for determining the function of a protein are costly, and thus the biochemical and the pharmaceutical enterprises need “in silico” methods for estimating if a protein posseses a specific function. As stated in the beginning of the introduction, one of the paradigms of structural biology is that the three dimensional structure of a protein determines its function. Even if recent researches on specific proteins called “intrinsically disordered” [17] show exceptions to this rule,

Table 1.2: Data growth in the SCOP database.

Scop version	PDB entries	Domains	Folds	Super-families	Families
1.75 (2009)	38221	110800	1195	1962	3902
1.73 (2007)	34494	97178	1086	1777	3464
1.71 (2005)	27599	75930	971	1589	3004
1.69 (2004)	25973	70859	945	1539	2845

In 2009, over the 60756 structures present in the PDB, 38221 were processed by SCOP. From these 38221 PDB structures, 110800 structural domains were extracted and then classified into 1195 folds, 1962 super-families and 3902 families.

the corresponding assumption that proteins sharing similar structures may share a common function was the starting point of the protein structure comparison.

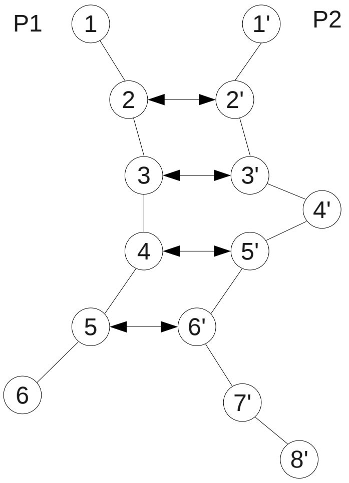
Earlier methods compared the proteins by using their sequences, like the Needleman-Wunsch algorithm [50] in 1970, the Smith-Waterman algorithm [61] in 1981, and more recently the BLAST heuristic [2] in 1990. BLAST, which allows comparison with a bank of sequences, is now the reference for the protein sequence comparison. Protein sequence comparison is still heavily used for two reasons. First, the corresponding algorithms are polynomial (they are based on dynamic programming or on polynomial heuristics) and are very fast. Second, the number of known protein sequences is clearly larger than the number of known protein structures. However, protein sequence comparison has a weak point, which consists in a “twilight zone”. Two protein sequences having less than about 30% of sequence identity (i.e. the same amino-acids at the same positions after alignment of sequences) are considered by sequence comparison to be unrelated, while such proteins can possess similar structures and functions [8]. This is the reason why protein structure comparison is considered to be more reliable than protein sequence comparison.

Unlike protein sequence comparison, there is not a clear consensus about how to compare two protein structures. However, we can easily describe the desired outputs of such comparisons. First, an optimal matching (also called alignment) between the amino-acids of the two proteins, according to matching rules which are dependant of the comparison method. This alignment represents what is common in the two protein structures. Second, a similarity score which is based on the optimal alignment. This score measures how similar are the two protein structures.

Definition 1 *Let P_1 and P_2 be two proteins, described by their ordered set of amino acids V_1 and V_2. Matching amino-acid i from V_1 with amino-acid k from V_2 is represented by the matching pair $i \leftrightarrow k$. An alignment is a sequence of matching pairs “$i_1 \leftrightarrow k_1$,..., $i_t \leftrightarrow k_t$”, as illustrated in figure 1.5. Since we focus on protein chains or on domains, we can make the assumption that this matching should be order preserving, as illustrated in* ***figure 1.6****.*

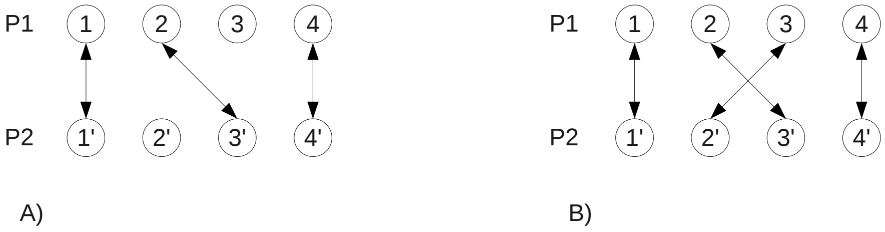
It is important to note that in most of the structure comparison methods, the contribution of a matching pair $i \leftrightarrow k$ to the objective function does not only depends on i and k, but also depends on the other chosen matching pairs. In such cases, as proved by lathrop [42], finding an optimal alignment is a NP-complete problem.

Figure 1.5: An example of alignment between two protein structures.



Between the two proteins P_1 and P_2, the alignment "$2 \leftrightarrow 2'$, $3 \leftrightarrow 3'$, $4 \leftrightarrow 5'$, $5 \leftrightarrow 6'$" is represented by the black arrows.

Figure 1.6: Order preserving matching.



A) Between P_1 and P_2, the alignment "$1 \leftrightarrow 1'$, $2 \leftrightarrow 3'$, $4 \leftrightarrow 4'$" is order preserving. B) Alignment "$1 \leftrightarrow 1'$, $2 \leftrightarrow 3'$, $3 \leftrightarrow 2'$, $4 \leftrightarrow 4'$" is not order preserving. It is sometimes referred as crossing matching.

Since the notion of similar three-dimensional structures is not clearly defined, a multitude of methods has been proposed and compared [27, 59, 60, 68], but no method has successfully imposed itself as a gold standard. It is not possible to give an exhaustive state of the art about the protein structure comparison methods, but the ones presented here illustrate the variety of the possible protein structure representations and of the similarity functions used for comparing the protein structures.

2.1 DALI

In DALI [35, 37], the structure of a protein is represented by a two-dimensional distance matrix, which contains the internal distances between the α-carbons of the amino-acids. DALI tries to find the alignment which minimize the differences between the sub-matrices corresponding the the matched amino-acids (i.e. which minimise the differences between the corresponding internal distances). More precisely, the contribution of matching pairs $i \leftrightarrow k$ and $j \leftrightarrow l$ to the objective function is :

$$score(ikjl) = 0.2 - \left(\frac{2 \times |d_{i,j} - d_{k,l}|}{d_{i,j} + d_{k,l}} \right) \times \exp \left(\frac{(d_{i,j} + d_{k,l})^2}{1600} \right), \qquad (1.1)$$

where $d_{i,j}$ (resp. $d_{k,l}$) is the euclidean distance between amino-acid i and j (resp k and l). Since finding such alignment is a NP-complete problem, DALI uses a heuristic solver based on the Monte-Carlo algorithm [48], which does not guaranty that the returned alignment is optimal.

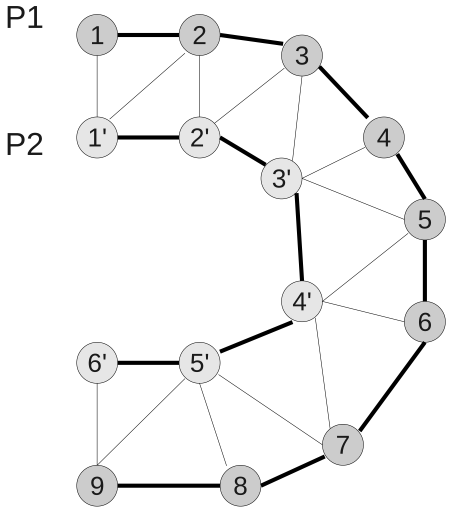
2.2 STRUCTAL

In STRUCTAL [24, 25, 64], the structure of a protein is represented by the three-dimensional coordinates of the amino-acid α-carbons. STRUCTAL tries to both find the transformation (which is a combination of a rotation and a translation) and the alignment which minimize the root-mean-squared-deviation (RMSD) of the coordinates between matched amino-acids (also called $RMSD_c$). Given (i) an alignment $A =$ "$i_1 \leftrightarrow k_1$,..., $i_n \leftrightarrow k_n$", and (ii) a transformation T such that after applying it, the coordinates of amino acid k from P_2 are $T(k)$, then the $RMSD_c$ of an alignment is :

$$RMSD_c(A, T) = \sqrt{\frac{\sum_{\forall i \leftrightarrow k \in A} (d_{i,T(k)})}{n}}, \qquad (1.2)$$

where n is the number of matching pairs in A. Since transformation T depends on all matching pairs, finding the alignment which minimise the $RMSD_c$ is also a NP-complete problem. In STRUCTAL, it is solved by using the following iterative algorithm. Each iteration is composed of two steps. First, according to an initial transformation T is computed an new optimal alignment A. Second, based on the new optimal alignment A, a new transformation T is computed and is given to the next iteration. This process is then repeated until convergence. Note however that this iterative algorithm is not an exact solver and thus does not guaranty that the returned alignment is optimal.

Figure 1.7: An example of triangulation between two proteins.



A triangulation consists in dividing the area in between the two proteins P_1 and P_2 into triangles which endpoints are the amino-acid α-carbons.

2.3 GAFIT

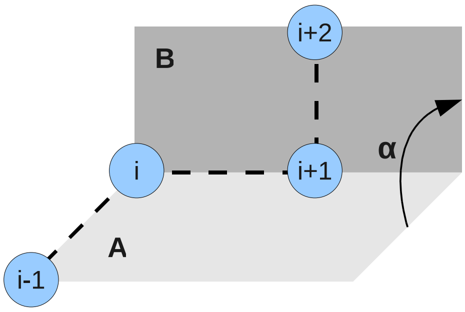
In principle, GAFIT [46] and STRUCTAL are very similar. Both try to find the transformation T and the alignment A which minimize the $RMSD$ of the coordinates between matched amino-acids. however, they differ in the algorithms they use. In GAFIT, a genetic algorithm [34] is used to find the transformation T. Evaluating the fitness of a member of the current population (i.e. how good is one possible transformation) is done by computing the $RMSD_c$ of the corresponding best alignment.

2.4 MINAREA

While STRUCTAL computes both an alignment A and a transformation T which both minimise the $RMSD_c$, MINAREA [18] computes both a triangulation (as illustrated in **figure 1.7**) between the amino-acid α-carbons and a transformation T which both minimise the surface area in-between the amino-acid α-carbons.

Since transformation T depends on the whole triangulation, minimising the surface in-between the amino-acid α-carbons is also a NP-complete problem. Similarly to STRUCTAL, this problem is solved by an iterative algorithm where each iteration is composed of two steps. First, given an initial transformation function T, a dynamic programming step is used to compute the triangulation leading to the smallest surface area. Second, based on this triangulation, a new transformation function T which minimises the surface area is computed and is given to the next iteration. This process is iterated until convergence. Again, this iterative algorithm is not an exact solver and thus does not guaranty that the returned triangulation is optimal.

Figure 1.8: A dihedral angle between four consecutive α-carbons.



The α-carbons $(i-1,\ i,\ i+1)$ define plane A, and the α-carbons $(i,\ i+1,\ i+2)$ define plane B. The angle between plane A and B is a dihedral angle (here named α).

2.5 VAST

VAST [26] (Vector Alignment Search Tool) is a hierarchical approach. In a first step, the protein structure is described by vectors which represent its secondary structure elements. VAST tries to find the alignment of vectors which optimize a probability function based on the RMSD (It is less probable, and thus more significant, to find a long alignment of secondary structure elements having a small RMSD value, compared to a random alignment having the same length). Finding this optimal alignment is modelled as looking for a clique in a graph and is solved by using the Bron and Kerbosch algorithm [9], which is an exact solver. Then, in a second step, VAST extends the optimal secondary structure alignment to the amino-acids by using a Gibbs-Sampling technique [23], which this time is not an exact solver, and thus, the returned amino-acids alignment may not be optimal.

2.6 YAKUSA

In YAKUSA [13], the protein structure is represented by sequences of dihedral angles between four consecutive amino-acid α-carbons (as illustrated in **figure 1.8**). The sequence does not contain exact angle values, but symbols corresponding to a discretization of all possible angle values (each symbol corresponding to an interval of 10 degrees). Comparing two protein sequences of dihedral angles (one called “query”, and the other one called “target”) is done in four steps.

1. The query is split in overlapping fixed size sub-sequences which are given to an automaton. This automaton both describes exact sub-sequences and similar sub-sequences (symbols are similar if they correspond to consecutive angle intervals, i.e. $[10^o, 20^o]$ is similar to $[20^o, 30^o]$).

2. The target sequence is scanned with the automaton in order to retrieve all similar short common sub-sequences, called “seeds”.

3. Seeds are extended to the longest possible SHSPs (Structural High Scoring Pair).

4. The best "compatible" SHSPs (according to a RMSD function) are assembled, and a score is computed for the query/target pair.

2.7 CMO

In CMO (Contact Map Overlap maximisation) [28], the protein structure is represented by an adjacency matrix, in which two amino-acids are adjacent if their euclidean distance in three-dimensional space is less than a given threshold (in such case, they are said to be in "contact"). CMO tries to find the alignment which maximises the number of common contacts (i.e. when two amino-acids which are in contact are matched with two amino-acids which are also in contact). In [11], and then in [69], CMO is associated to scoring schemes which are efficient for comparing two protein structures. The best one, from [69] is :

$$Sim(P_1, P_2) = \frac{2 \times ncc}{nc1 + nc2}, \tag{1.3}$$

where ncc is the number of common contacts between P_1 and P_2, $nc1$ and $nc2$ are the number of contacts in P_1 and P_2. CMO is a NP-complete problem [29] for which many algorithms have been proposed [11, 12, 14, 63, 69]. However, despite its qualities, efficiently solving CMO is still an open problem. Conceiving an efficient solver for CMO which guarantees the quality of the obtained results is the main objective of this thesis.

3 The alignment graph approach

In this thesis, we propose a general canvas for modelling the protein structure comparison which is based on a specific graphs that we called "alignment graphs" and whose particular structure allows the development of efficient solvers. Let us first introduce some notations and basic graph theory knowledge.

3.1 Undirected graph and cliques problems

An undirected graph is usually denoted by $G = (V, E)$, where V is the set of vertices and E is the set of edges. Two vertices i and j are said to be adjacent if they are connected by an edge of E. A clique of a graph is a subset of its vertex set, such that any two distinct vertices in it are adjacent.

Definition 2 *The **maximum clique problem**, also called maximum cardinality clique problem, (MCC) is to find a largest, in terms of vertices, clique of an arbitrary undirected graph G, which will be denoted by $MCC(G)$.*

The maximum clique problem is one of the first problem shown to be NP-Complete [40] and it has been studied extensively in the literature. Interested readers can refer to [7] for a detailed state of the art about the maximum clique problem.

When weights are associated to the vertices and to the edges of G, then other maximum clique related problems arise.

Definition 3 *The* Maximum Vertex Weighted clique problem *(MVW) consists in finding in G a clique with a maximum sum of vertex weights.*

If the vertex weights are all equal to 1, then MVW is equivalent to MCC. Since MCC is a special case of MVW, then MVW is also NP-Complete.

Definition 4 *The* Maximum Edge Weighted clique problem *(MEW) consists in finding in G a clique with a maximum sum of edge weights.*

If the edge weights are all equal to 1, then MEW is equivalent to MCC, so MEW is also NP-Complete. All these clique problems have been intensively investigated [1, 7, 9, 10, 58].

Definition 5 *The* Maximum Weighted Clique problem *(MWC) consists in finding in G a clique having a maximum sum of vertex and edge weights.*

It is easily seen that MCC, MVW and MEW are all special cases of MWC. If the vertex weights are equal to zero, then MWC is equivalent to MEW, if edge weights are all equal to zero, then MWC is equivalent to MVW. If the vertex weights are all equal to 1 and the edge weights are all equal to zero, then MWC is equivalent to MCC. Thus, MWC is also NP-Complete.

3.2 Alignment graphs

In this thesis, we focus on specific k-partite graphs, which we define as follows.

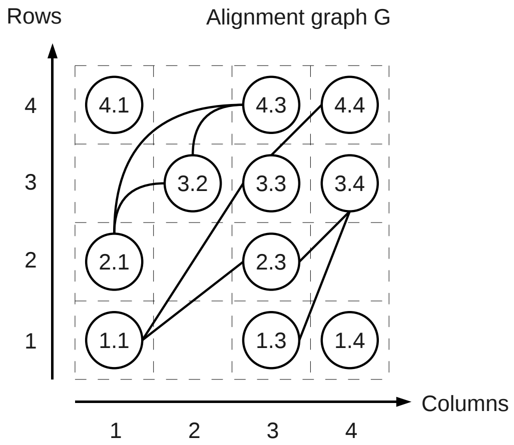
Definition 6 *A $m \times n$* ***alignment graph*** *$G = (V, E)$ is a graph in which the vertex set V is depicted by a (m-rows) $\times$ (n-columns) array T, where each cell $T[i][k]$ contains at most one vertex $i.k$ from V (note that for both arrays and vertices, the first index stands for the row number, and the second for the column number). Two vertices $i.k$ and $j.l$ can be connected by an edge $(i.k, j.l) \in E$ only if $i < j$ and $k < l$. An example of such alignment graph is given in* ***figure 1.9****.*

It is easily seen that the m rows form a m-partition of the alignment graph G, and that the n columns also form a n-partition. In the rest of this paper we will use the following notations. A successor of a vertex $i.k \in V$ is an element of the set $\Gamma^+(i.k) = \{j.l \in V \text{ s.t. } (i.k, j.l) \in E, i < j \text{ and } k < l\}$. $V^{i.k}$ is the subset of V restricted to vertices in rows j, $i \leq j \leq m$, and in columns l, $k \leq l \leq n$. Note that $\Gamma^+(i.k) \subset V^{i+1.k+1}$. $G^{i.k}$ is the subgraph of G induced by the vertices in $V^{i.k}$. The cardinality of a vertex set U is $|U|$.

3.3 Relations with protein structure similarity

From a general point of view, two proteins P_1 and P_2 can be represented by their ordered set of components N_1 and N_2, and estimating their similarity can be done by finding the longest order-preserving matching (also called alignment) between the elements of N_1 and N_2. In our approach, such matchings are represented in a $|N_1| \times |N_2|$ alignment graph $G = (V, E)$, where each row corresponds to an element of N_1 and each column corresponds to an element of N_2. A vertex $i.k$ is in V (i.e.

Figure 1.9: A 4 × 4 alignment graph.



matching $i \leftrightarrow k$ is possible), only if element $i \in N_1$ and $k \in N_2$ are compatible. An edge $(i.k, j.l)$ is in E if and only if (i) $i < j$ and $k < l$, for order preserving, and (ii) matching $i \leftrightarrow k$ is compatible with matching $j \leftrightarrow l$.

There is a multitude of alignment methods and they differ mainly by :

1. The nature of the elements of N_1 and N_2.

2. The compatibility definitions between elements (for creating vertex $i.k$) and between couples of matching pairs of elements (for creating edges $(i.k, j.l)$).

3. The subset of V corresponding to an optimal matching between P_1 and P_2.

If we are looking for an alignment such that all couples of matched pairs of amino-acids are compatible, then we are looking for a clique in G .

Many protein structure similarity related problems from the literature can be converted into clique problems in alignment graphs, as we illustrate here with DALI, with the secondary structure alignment in VAST [26], and the Contact Map Overlap Maximization problem (CMO) [28].

Dali: When the two proteins P_1 and P_2 are represented by their ordered sets of amino-acids N_1 and N_2, Dali, which is based on distance matrices, can be modelled in an $|N_1| \times |N_2|$ alignment graph in the following way. All amino-acids from P_1 are compatible with all amino-acids from P_2, and thus, for all $i \in N_1$ and all $k \in N_2$, vertex $i.k$ is in V. User can choose if the alignment is order preserving or not : either all matchings $i \leftrightarrow k$ are compatible with matchings $j \leftrightarrow l$, or $i \leftrightarrow k$ is compatible with $j \leftrightarrow l$ only if $i < j$ and $k < l$. If $i \leftrightarrow k$ is compatible with $j \leftrightarrow l$ then edge $(i.k, j.l)$ is in E. Dali can use two different scoring schemes. The first one, called "rigid" associates to each edge $(i.k, j.l) \in E$ a weight $w^r_{ikjl} = \tau - |d_{i,j} - d_{k,l}|$, where τ is a rigid distance threshold, fixed to 1.5 Å. The second one, called "elastic", associates a weight $w^e_{ikjl} = \tau - \left(\frac{2 \times |d_{i,j} - d_{k,l}|}{d_{i,j} + d_{k,l}}\right) \times \exp\left(\frac{(d_{i,j} + d_{k,l})^2}{1600}\right)$, where τ is a relative distance threshold, fixed to 0.2. The best alignment is then a clique with maximum sum of (either rigid or elastic) edge weights.

VAST: In VAST, N_1 and N_2 contain 3D vectors representing the secondary structure elements (SSE) of P_1 and P_2. Matching $i \leftrightarrow k$ is possible if vectors i and k have similar norms and correspond either both to α-helices or both to β-strands. Finally, matching $i \leftrightarrow k$ is compatible with matching $j \leftrightarrow l$ only if the couple of vectors (i, j) from P_1 can be well superimposed in 3D-space with the couple of vectors (k, l) from P_2.

CMO: In the next chapter of this thesis, we model CMO in alignment graph, but not by using cliques. However, a maximum clique formulation in alignment graphs was proposed by Strickland et al. in [63] and then reused in [56], but they did not exploit the properties of the alignment graphs.

4 Integer programming and Lagrangian relaxation

In this thesis, we solve the protein structure comparison problems by using integer programming formulations and Lagrangian relaxation. To better understand the rest of the thesis, let us introduce these two combinatorial optimisation techniques. The interested readers should refer to [30] for more detailled explanations about relaxation and Lagrangian relaxation. Please note that the notions and definitions presented in this section are given in the context of the maximisation of a function $f(x)$. The case of the minimisation is not presented, since minimising $f(x)$ is equivalent to maximising $-f(x)$.

4.1 Preliminary notations and definitions

The problem P of maximising a given function $f(x)$ over the set V of the feasible x variables is denoted by:

$$P = \max\{\ f(x)\ |\ x \in V\ \}. \tag{1.4}$$

The optimal value of P is denoted by $v(P)$, and the corresponding solution in terms of x variables is denoted by $z(P)$.

According to Geoffrion [22], the relaxation of a maximisation problem is defined as follows :

Definition 7 *The problem $RP = \max\{\ g(x)\ |\ x \in W\ \}$ is* ***a relaxation*** *of the problem $P = \max\{\ f(x)\ |\ x \in V\ \}$ if and only if :*

- *The feasible set of RP contains the feasible set of P (i.e., $V \subset W$).*
- *Over the feasible set of P, the objective function of RP dominates the objective function of P (i.e., $\forall x \in V, f(x) \leq g(x)$).*

The most important relation between the original and the relaxed problem is that $v(RP)$ is an upper-bound on $v(P)$ (i.e., $v(RP) \geq v(P)$). On the other hand, the relaxed solution $x = z(RP)$, which is usually infeasible for P, can often be repaired into a feasible solution x' which then provides a lower-bound on $v(P)$ (i.e., $f(x') \leq v(P)$).

The relaxation is widely used for generating bounds in branch and bound algorithms. Towards this goal, the relaxation scheme should have the following properties. First, the difference $v(RP) - v(P)$, called gap, should be as small as possible. Second, the relaxed problem should be easy to solve (i.e., in polynomial times).

4.2 Integer programming

The Integer Programming (IP) approach consists in reformulating a given problem P as a maximisation of a linear function $f(x)$, where the variables are integers ($x \in \mathbb{Z}$), and subjected to linear constraints $Ax \leq b$.

$$P = \max\{\ f(x) \mid x \in \mathbb{Z},\ Ax \leq b\ \}. \tag{1.5}$$

The most widely used relaxation scheme in the context of integer programming is the continuous relaxation, which consist in removing (relaxing) the integrality constraints on the variables ($x \in \mathbb{Z}$ becomes $x \in \mathbb{R}$). The corresponding relaxed problem is the Linear Programming problem

$$LP = \max\{\ f(x) \mid x \in \mathbb{R},\ Ax \leq b\ \} \tag{1.6}$$

which can be solve (for example) by using the simplex algorithm (G. Dantzig, 1947)

4.3 Lagrangian relaxation

The Lagrangian relaxation was proposed by Held and Karp in 1970 [32], and is based on the observation that given a NP-Complete IP problem $P = \max\{\ f(x) \mid x \in \mathbb{Z},\ Ax \leq b\ \}$, the set of constraints $Ax \leq b$ can be divided into two sets, $A'x \leq b'$ and $A''x \leq b''$, such that when subjected to both sets of constraints, P is hard to solve (NP-Complete), while relaxing (removing) the constraints $A''x \leq b''$ makes P easy to solve (in polynomial times). The constraints $A''x \leq b''$ are then called "complicating" constraints.

Definition 8 *The **Lagrangian relaxation** of the problem $P = \max\{\ f(x) \mid x \in \mathbb{Z},\ A'x \leq b',\ A''x \leq b''\ \}$, with nonnegative Lagrangian multipliers λ associated to the complicating constraints $A''x \leq b''$, is the new problem*

$$LR(\lambda) = \max\{\ f(x) + \lambda(b'' - A''x) \mid x \in \mathbb{Z},\ A'x \leq b'\ \}. \tag{1.7}$$

In the Lagrangian relaxation, the complicating constraints are removed from the set of constraints, and are then introduced as penalties into the objective function associated with nonnegative multipliers λ. Among the many relations between P and $LR(\lambda)$, the most interesting ones are :

- $LR(\lambda)$ is also an IP problem, but easier to solve than P.
- $LR(\lambda)$ is a relaxation of P for *any* values of λ, and consequently, $v(P) \leq v(LR(\lambda))$, for any values of λ.
- $LR(\lambda)$ is a tighter relaxation of P than LP for any values of λ, i.e., $v(P) \leq v(LR(\lambda)) \leq v(LP)$.

4.4 Lagrangian dual problem

Definition 9 *The **Lagrangian Dual** problem of P, relatively to the complicating constraints $A''x \leq b''$, is the problem*

$$LD = \min_{\lambda} \{ \ v(LR(\lambda)) \mid \lambda \in \mathbb{R}^{+} \ \}, \tag{1.8}$$

which consists in finding the smallest upper-bounds of $v(P)$ given by $v(LR(\lambda))$.

In order to tighten the bounds of P given by $LR(\lambda)$, or eventually to solve P, we need to solve (or at least to find a good sub-optimal solution) the Lagrangian dual problem of P.

Chapter 2

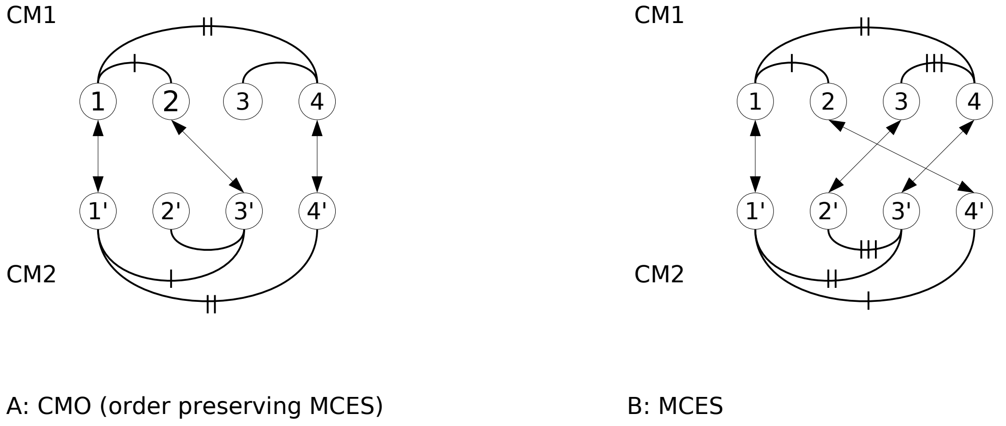
The Contact Map Overlap maximization problem: from mathematical model to efficient exact solver

1 Introduction to the Contact Map Overlap maximisation problem

The Contact Map Overlap maximization problem (CMO), is a protein structure comparison scoring scheme first proposed by [28]. This measure is robust, takes partial matching into account, is translation-invariant and it captures the intuitive notion of similarity very well. The protein primary structure is the linear arrangement of its amino-acids. Under specific physiological conditions, this linear arrangement will fold and adopt a complex 3D shape, called tertiary structure. In this folded state, amino-acids that are far away along the linear arrangement may come into proximity in 3D space and form contacts. This proximity relation is captured by a contact map. The *contact map* of a protein P is a simple graph with vertices corresponding to the amino-acids of P and where a contact edge (i, j) between two amino-acids i and j exists if and only if their Euclidian distance in the protein fold is smaller than a given threshold.

In the CMO approach one tries to evaluate the similarity between two proteins by determining the maximum overlap of their contact maps. The formal definition is as follows. Let $I = (i_1, i_2, \dots i_s), i_1 < i_2 < \dots i_s$ and $J = (j_1, j_2, \dots j_s), j_1 < j_2 < \dots j_s$ be arbitrary subsets of vertices from the first and second contact maps, respectively. Under the alignment (matching, one-to-one map) $i_k \longleftrightarrow j_k, k = 1, 2, \dots s$, the edge (k, l) is *common* (an overlap occurs) if and only if both edges (i_k, i_l) and (j_k, j_l) exist. The CMO problem is to find the optimal I and J, where optimality means maximum number of common edges. Since the sets of amino-acids are supposed linearly ordered then for any two sets I and J of the same cardinality the only one-to-one map that respects this order (*non-crossing matching*) is the one given above. Problems with the same goal (maximum number of common edges) but without restriction on the mapping (i.e. crossing matchings are allowed) are known as *Maximum Common Subgraph* (MCS) [21] and *Maximum Common Edge Subgraph*

Figure 2.1: Comparing CMO with MCES.



A: The matching "$1 \leftrightarrow 1', 2 \leftrightarrow 3', 4 \leftrightarrow 4'$", represented by the arrows, matchs vertex set $\{1,2,4\}$ from CM1 with vertex set $\{1',3',4'\}$ from CM2. The corresponding common edges (two in this case) are denoted by the same number of vertical dashes. It is an order preserving MCES with maximum score 2. **B:** Vertex set $\{1,2,3,4\}$ from CM1 is matched with vertex set $\{1',4',2',3'\}$ from CM2. This is a MCES with score 3. The order is not preserved.

(MCES) [47]. The later notion is usually used in the context of chemical structure similarity. In fact in both problems, in CMO as well as in and MCES, one looks for the optimal I and J. However, while for CMO finding these sets merely solves the problem, discovering MCES requires solving MCS for the graphs induced by I and J respectively. **Figure 2.1** illustrates these notions.

Thus, CMO is just a way to introduce a similarity measure, a function $\rho(A, B)$ that associates a numerical value with a pair of contact maps (A, B). Often this is normalized to be in the interval $[0, 1]$. CMO is an NP-hard problem [29] (it is easily seen that if one of the contact maps is a complete graph of k vertices then the CMO is equivalent to the decision problem of the existence of a clique of cardinality k in a given graph), and thus designing efficient algorithms that guarantee the CMO quality is a problem that has eluded researchers so far. For a detailed state of the art on this subject the interested reader can refer to a recent paper by [69]. Here we focus on designing exact algorithm for solving CMO problems. Towards this goal, [14] were the first to tackle maximum CMO by integer programming. This approach seems promising, especially when coupled with Lagrangian relaxation [11, 12]. Alternative approaches have been recently proposed by [69], who proposed a reduction technique in the context of Branch and Bound (B&B), as well by [63], who reformulated CMO as a maximum clique problem. This chapter demonstrates once more the efficiency of Lagrangian relaxation approach when applied to maximum CMO. Our interest in CMO was provoked by its resemblance to the *Protein Threading Problem* (PTP) for which Andonov et al. have presented a methodology based on the non-crossing matching in bipartite graphs [3]. It yielded a highly efficient algorithm by using Lagrangian duality [70]. We aim to extend this approach to CMO by presenting it as a matching problem in a bipartite graph, which in turn will be posed as a maximum weight augmented path in a structured graph.

The contributions of this chapter are as follows. We propose a new mixed integer programming (MIP) formulation for the CMO problem. We present an efficient Lagrangian relaxation to solve our model and incorporate it into a Branch and Bound (B&B) search. On data set available from the literature we observe that our algorithm is faster than the four above mentioned exact algorithms [11, 14, 63, 69], and is even faster than two recent heuristics [38, 54]. The comparisons with the most recent algorithms [11, 69] were performed on a benchmark which is widely used in the CMO community (the Skolnick set), and we noted that our solver outperforms them significantly, both in time and in quality of the provided bounds. We also solve new hard Skolnick set instances. Finally, we used our method as a classifier on both the Skolnick set and the Proteus_300 set (a large benchmark of 300 domains that we extracted from SCOP [49]). We are not aware of any previous attempt to apply a CMO approach on such a large database. The obtained results are in perfect agreement with SCOP classification and clearly demonstrate that our algorithm can be used as a tool for large scale classification.

2 Integer programming model

Let us first introduce some notation. The contact maps of two proteins P1 and P2 are given by graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $n_1 =\mid V_1 \mid$ and $n_2 =\mid V_2 \mid$. The vertices in V_1 and in V_2 correspond to the amino-acids of the proteins and it is convenient to see them as ordered points on a line. The edges in E_1 and in E_2 correspond to the contacts. The right and left neighbors of vertex i are elements of the sets $\delta^+_m(i) = \{j | j > i, (i, j) \in E_m\}$ and $\delta^-_m(i) = \{j | j < i, (j, i) \in E_m\}$, for $m \in \{1, 2\}$. Let $i \in V_1$ be matched with $k \in V_2$ (i.e. $i \leftrightarrow k$) and $j \in V_1$ be matched with $l \in V_2$ (i.e $j \leftrightarrow l$). We will call a matching *non-crossing* if $i < j$ implies $k < l$. Feasible alignments of two proteins P_1 and P_2 are given by non-crossing matchings in the complete bipartite graph B with a vertex set $V_1 \cup V_2$ (see **figure 2.2**: Left).

Let the weight w_{ikjl} associated to the couple of matching pairs $(i \leftrightarrow k, j \leftrightarrow l)$ be set as follows :

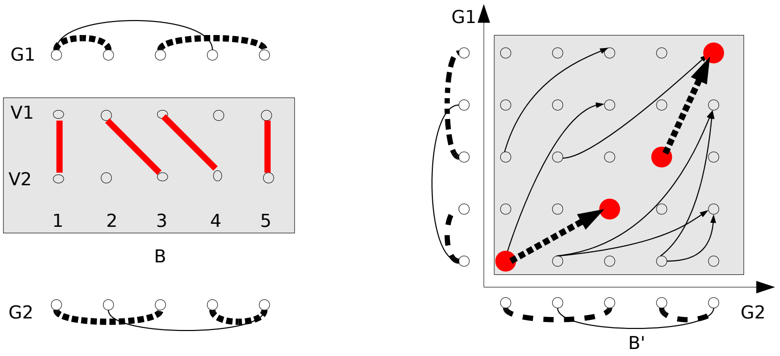
$$w_{ikjl} = \begin{cases} 1 & \text{if } (i, j) \in E_1 \text{ and } (k, l) \in E_2 \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

For a given non-crossing matching M in B we define its weight $w(M)$ as the sum of weights over all pairs of edges in M. CMO consists then in maximizing $w(M)$, where M belongs to the set of all non-crossing matchings in B.

In [3, 70], Andonov et al. have already dealt with similar non-crossing matchings and have proposed for them a network flow presentation. This approach is adapted to CMO as follows (see **figure 2.2**). The edges of the bipartite graph B are mapped to the vertices of a $n_1 \times n_2$ alignment graph $B' = (V', E')$ according to the rule: a vertex $i.k \in V'$ corresponds to the edge (i, k) in B and vice versa. We also add arcs $(i.k, j.l) \in E'$ iff $w_{ikjl} = 1$.

Definition 10 *An* ***increasing subset of vertices in an alignment graph*** B' *is an arbitrary ordered set* $\{i_1.k_1,\ i_2.k_2,\ \ldots,\ i_t.k_t\}$ *of vertices in* B' *such that* $i_j < i_{j+1}$ *and* $k_j < k_{j+1}$ *for* $j \in [1, t-1]$.

Figure 2.2: Relationship between a matching in a bipartite graph B and an increasing subset of vertices in the corresponding alignment graph B'



Left: Two contact maps $G1$ and $G2$, and a matching in the bipartite graph B (in the grey area). Note that B is a complete graph, but for the sake of simplicity only the edges of the considered matching ($M = \{(1,1)(2,3),(3,4),(5,5)\}$) are visualized. According to (2.1), $w(M) = 2$. **Right**: The same matching is visualized in the grid alike graph B' as the increasing subset of vertices $\{1.1, 2.3, 3.4, 5.5\}$. It activates the arcs (1.1, 2.3) and (3.4, 5.5). The score of the path is the number of these arcs (i.e. 2 in this case).

The correspondence between an increasing subset of vertices and a non-crossing matching is then obvious. Searching for feasible alignments of two proteins is thus converted to searching for an increasing subset of vertices. In B', solving CMO, corresponds to finding the densest (in terms of arcs) subgraph of B' whose vertex set is an increasing subset of vertices. To each vertex $i.k \in V'$ we now associate a 0/1 variable x_{ik}, and to each arc $(i.k, j.l) \in E'$, a 0/1 variable y_{ikjl}. Denote by X the set of increasing subsets of vertices. The problem can be stated as follows :

$$v(CMO) = \max \sum_{(i.k,j.l)\in E'} y_{ikjl} \tag{2.2}$$

subject to

$$x_{ik} \geq \sum_{l\in\delta_2^+(k)} y_{ikjl}, \qquad j \in \delta_1^+(i), \quad i \in [1, n_1 - 1], \quad k \in [1, n_2 - 1]. \tag{2.3}$$

$$x_{ik} \geq \sum_{j\in\delta_1^+(i)} y_{ikjl}, \qquad l \in \delta_2^+(k), \quad i \in [1, n_1 - 1], \quad k \in [1, n_2 - 1]. \tag{2.4}$$

$$x_{ik} \geq \sum_{l\in\delta_2^-(k)} y_{jlik}, \qquad j \in \delta_1^-(i), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{2.5}$$

$$x_{ik} \geq \sum_{j\in\delta_1^-(i)} y_{jlik}, \qquad l \in \delta_2^-(k), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{2.6}$$

$$x \in X \tag{2.7}$$

Actually, we know how to represent X with linear constraints. It is easily seen that definition 10 of a feasible path yields the following inequalities

$$\sum_{l=1}^{k} x_{il} + \sum_{j=1}^{i-1} x_{jk} \leq 1, \qquad i \in [1, n_1], \quad k \in [1, n_2]. \tag{2.8}$$

The same definition also implies that the j-th amino-acid from $P1$ could be matched with at most one amino-acid from $P2$ and vice-versa. This explains the sums on the right hand sides of (2.3) and (2.4) (for arcs having their tails at vertex $i.k$); and (2.5) and (2.6) (for arcs heading to $i.k$). Any arc $(i.k, j.l)$ can be activated (i.e. $y_{ikjl} = 1$) iff $x_{ik} = 1$ and $x_{jl} = 1$ and in this case the respective constraints are active because of the objective function.

A tighter description of the polytope defined by (2.3)–(2.6) and $0 \leq x_{ik} \leq 1$, $0 \leq y_{ikjl}$ could be obtained by lifting the constraints (2.5) and (2.6) as shown in **figure 2.3**. The gray area contains the predecessors of the vertex $i.k$ in the graph B' and they form a grid of $\delta_1^-(i)$ rows and $\delta_2^-(k)$ columns. Let $i_1, i_2, \ldots, i_s$ be all the vertices in $\delta_1^-(i)$ ordered according to the numbering of the vertices in V_1 and likewise $k_1, k_2, \ldots, k_t$ in $\delta_2^-(k)$. Then the vertices in the l-th column $\{i_1.k_l, i_2.k_l, \ldots i_s.k_l\}$ correspond to pairwise crossing matchings and at most one of them could be chosen in any feasible solution $x \in X$ (see constraint (2.6)). This "all crossing" property holds even if we add to this set the following two sets: $\{i_s.k_1, i_s.k_2, \ldots, i_s.k_{l-1}\}$ and $\{i_1.k_{l+1},\ i_1.k_{l+2},\ \ldots,\ i_1.k_t\}$. Denote by $col_{ik}(l)$ the union of these three sets, and analogously, by $row_{ik}(j)$ the corresponding union for the j-th row of the grid ($col_{ik}(l)$ (resp.$row_{ik}(j)$) is illustrated in figure 2.3 b (resp. d)), . When the grid is one column/row, the set $col_{ik}(l)/row_{ik}(j)$ reduces to this column/row only.

Now a tighter LP relaxation of (2.3)–(2.6) is obtained by substituting (2.5) with (2.9), and (2.6) with (2.10).

$$x_{ik} \geq \sum_{(r,s) \in row_{ik}(j)} y_{rsik}, \qquad j \in \delta_1^-(i), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{2.9}$$

$$x_{ik} \geq \sum_{(r,s) \in col_{ik}(l)} y_{rsik}, \qquad l \in \delta_2^-(k), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{2.10}$$

Remark: Since we are going to apply the Lagrangian technique there is no need for either an explicit description of the set X or for lifting the constraints (2.3) and (2.4).

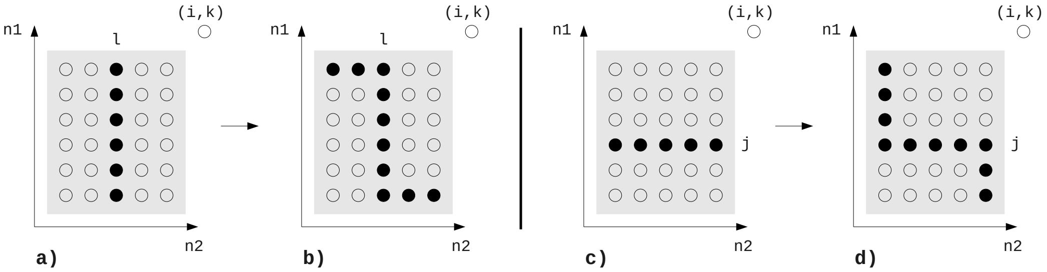
3 The lagrangian relaxation based solver

Here, we show how the Lagrangian relaxation of constraints (2.9) and (2.10) leads to an efficiently solvable problem, yielding upper and lower bounds that are generally better than those found by the previously published algorithms [11, 69].

3.1 Lagrangian relaxation approach

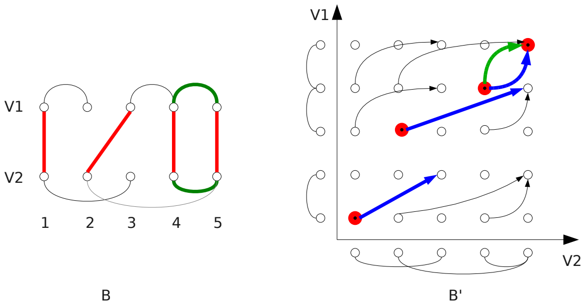
Towards this goal we first add to problem (2.2)-(2.8) the following constraint.

Figure 2.3: Illustration of the tighter LP model



The shadowed area represents the set of vertices in V' which are tails for the arcs heading to a given point $i.k$. The boldfaced points in a) visualize the set $\delta_1^-(i)$ for a given l and illustrate (2.6); b) depicts $col_{ik}(l)$ in the tightened constraint (2.10). The boldfaced points in c) visualize the set $\delta_2^-(k)$ for a given j and illustrate (2.5); d) depicts $row_{ik}(j)$ in the tightened constraint (2.9).

Figure 2.4: Relaxed problem.



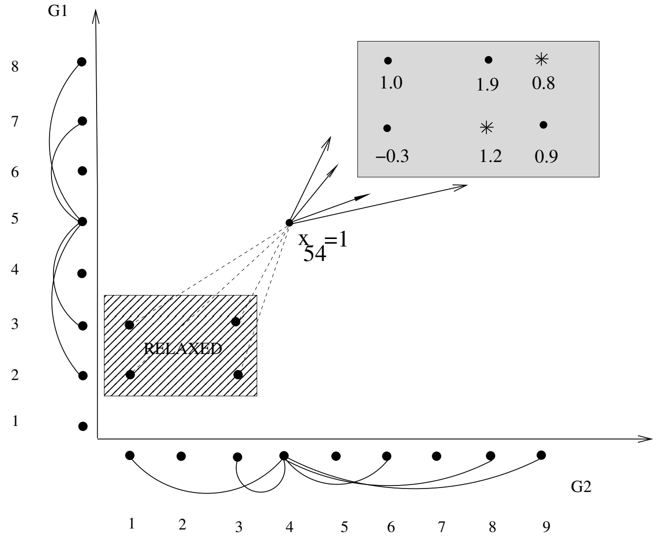
Left: Graph B represents a matching (in red lines) yielding one common contact (in green). Right: The same matching is represented as an increasing subset of vertices with four red nodes in the alignment graph B'. When the constraints related to the incoming arcs are relaxed, the nodes incident to the heads of these arcs may not be activated. Arcs activating the node incident to their tail only (in blue) give an upperbound (UB=3). Arcs activating their both extremities (in green) provide a lowerbound (LB=1). The exact solution here is 2.

$$\sum_{l\in\delta_2^+(b),l\leq k} y_{abil}+\sum_{j\in\delta_1^+(a),j<i} y_{abjl}\leq 1, a\in[1,n_1], b\in[1,n_2], i\in\delta_1^+(a), k\in\delta_2^+(b) \quad (2.11)$$

Constraint (2.11) requires that the heads of all arcs outgoing from a vertex $a.b$ must satisfy the definition of an increasing subset of vertices. It is in fact induced by the constraint (2.8) and the new problem is therefore equivalent to the original one. We now relax constraints (2.9) and (2.10), authorizing in this way the heads of the activated axes not to be incident with the chosen nodes of the increasing subset of vertices. Activated arcs having only their tail lying on the increasing subset of vertices provide then an upper bound of the original problem, while arcs having their both extremities lying on the increasing subset of vertices give a lower bound. An illustration is given on **figure 2.4**.

Let $\lambda_{ikj}^h \geq 0$ (respectively $\lambda_{ikj}^v \geq 0$) be a Lagrangian multiplier assigned to each

Figure 2.5: The sets of left/right neighbors of a given vertex



The boxes below/above vertex 5.4 contains its left/right neighbors. The dashed lines symbolize the relaxed constraints (2.9) and (2.10) for $i.k = 5.4$. The numbers on its right neighbors correspond to the values of the coefficients $c_{54jl}(\lambda)$ for a given λ. The solution of the local problem in 5.4 equals 2 and is given by vertices 7.8 and 8.9 (indicated by stars). This value is used in the global problem.

constraint (2.9) (respectively (2.10)). By adding the slacks of these constraints to the objective function with weights λ, we obtain the Lagrangian relaxation of the CMO problem

$$\begin{aligned} LR(\lambda) = \quad \max \sum_{(i.k,j.l)\in E'} y_{ikjl} \;&+ \sum_{i,k,j\in\delta_1^-(i)} \lambda_{ikj}^h (x_{ik} - \sum_{(r,s)\in row_{ik}(j)} y_{rsik}) \\ &+ \sum_{i,k,l\in\delta_2^-(k)} \lambda_{ikl}^v (x_{ik} - \sum_{(r,s)\in col_{ik}(l)} y_{rsik}) \end{aligned} \tag{2.12}$$

subject to $x \in X$, (2.3), (2.4) and $y \geq 0$. Now, we are going to show the polynomial (linear) complexity of this problem. The proof is constructive and it is used in the algorithm described in 3.3.

Theorem 1 *$LR(\lambda)$ can be solved in $O(|V'| + |E'|)$ time.*

Proof: The proof is better seen as a call to two kinds of optimization problems, one is called local and the other one global, which are both solved using the same dynamic programming approach.

Local problem :

To each edge $(i.k, j.l) \in E'$, we associate a weight $c_{ikjl}(\lambda) = 1 - \lambda_{ikj}^h - \lambda_{ikl}^v$, which corresponds to the coefficient value of y_{ikjl} in (2.12). The local problem is, for each vertex $i.k \in V'$, to find the best subset of its outgoing edges (i.e., to find the values of the corresponding y_{ikjl} variables, $j \in \delta_1^+(i), l \in \delta_2^+(k)$) such that :

- all activated edges have their heads lying on an increasing subset of vertices,
- the sum of their weights, denoted by $c_{ik}(\lambda)$, is maximal.

All arcs in E' outgoing from $i.k$ can be put in one-to-one correspondence with the entries of a $|\delta_1^+(i)| \times |\delta_2^+(k)|$ array denoted by t_{ik}. To each entry $t_{ik}(j,l)$ in this array, we assign the profit $c_{ikjl}(\lambda)$. Then, the local problem is equivalent to searching in the array t_{ik} for a maximum weighted increasing subset of vertices (i.e : a subset of cells which contains vertices such that they form an increasing subset of vertices, and such that the sum of their weights is maximum.

Global problem :

To each vertex $i.k \in V'$, we associate a weight $c_{ik}(\lambda) + \sum_{j \in \delta_1^-(i)} \lambda_{ikj}^h + \sum_{l \in \delta_2^-(k)} \lambda_{ikl}^v$ (where the last two terms are the coefficients of x_{ik} in (2.12)). The global problem is then to find the values of the x_{ik} variables such that :

- all activated vertices lie on an increasing subset of vertices,
- the sum of their weights is maximal.

Again, all vertices in V' can be put in one-to-one correspondence with the entries of a $|V_1| \times |V_2|$ array denoted T_g, and for all entries in this array, we set $T_g(i,k) = c_{ik}(\lambda) + \sum_{j \in \delta_1^-(i)} \lambda_{ikj}^h + \sum_{l \in \delta_2^-(k)} \lambda_{ikl}^v$. The global problem is then equivalent to searching in T_g for a subset of cells corresponding to a maximum weighted increasing subset of vertices. See **figure 2.5** for an attempt to visualize parts of this proof. Note that because of relaxation some y_{ikjl} variables could be equal to 1 in the optimal solution of $LR(\lambda)$ even though if $x_{jl} = 0$.

Dynamic programming approach :

In this manner, for the both above problems, a 2D array is created. The definition (10) implies that finding a maximum weighted increasing subset of vertices in a $n \times m$ array T can be done by the following dynamic programming (DP) recurrence. Let T_s be an array which keeps the value of a maximum weighted increasing subset of vertices up to the cell $T(i,k)$. Then :

$$T_s(i,k) = \begin{cases} 0 & \text{if } i = 0 \text{ or } k = 0 \\ \max \begin{pmatrix} T_s(i,k-1),\ T_s(i-1,k), \\ T_s(i-1,k-1) + T(i,k) \end{pmatrix} & \text{otherwise.} \end{cases} \tag{2.13}$$

We are looking for the value $T_s(n,m)$, which can be computed in $O(n \times m)$ time complexity. We implemented the DP recurrence as presented in **algorithm 1**. Note that a second pass of dynamic programming is needed in order to retrieve the cells participating in this maximum weighted increasing subset of vertices.

Thus, $c_{ik}(\lambda)$ are computed by calling the DP algorithm on the corresponding array t_{ik} and it is done in $O(|\delta_1^+(i)| \times |\delta_2^+(k)|)$ time. The sum over all local DP calls (one for each $(i,k) \in V'$) gives a $O(|E'|)$ time complexity. Then, we can find the (global) optimal solution to $LR(\lambda)$ by calling the DP algorithm on the array T_g. This is done in $O(|V'|)$ time complexity. This gives us a total time complexity of $O(|V'| + |E'|)$ for solving $LR(\lambda)$.

Algorithm 1 DP(T), where T is a table containing n rows and m columns.

```
Require: Let $T_s$ and $T_m$ be two arrays of $(n+1)$ rows and $(m+1)$ columns, used for
    storing partial results. $T_s$ keeps the optimal sum of profit for cells up to $(i,k)$,
    and $T_m$ keeps the corresponding selected cells. Let $L$ be the maximum weighted
    increasing subset of vertices in $T$, and $Best\_Sum$ be the corresponding sum of
    weights.

# Initialisation
Column 0 and row 0 of $T_s$ are initialized with 0.
$L \leftarrow \{\emptyset\}$.
$Best\_Sum = 0$.

# Forward Step : Find optimal sum of profits.
for $col = 1$ to $m$ do
    for $row = 1$ to $n$ do
        if $T_s(row, col-1) > T_s(row-1, col)$ then
            $T_s(row, col) = T_s(row, col-1)$.
            $T_m(row, col)$ = 'left'.
        else
            $T_s(row, col) = T_s(row-1, col)$.
            $T_m(row, col)$ = 'down'.
        end if
        if $T(row, col) + T_s(row-1, col-1) > T_s(row, col)$ then
            $T_s(row, col) = T(row, col) + T_s(row-1, col-1)$.
            $T_m(row, col)$ = 'diag'.
        end if
    end for
end for
$Best\_Sum = T_s(n, m)$.

# Backward Step : Retrieve corresponding cells.
$col = m, row = n$.
while $col \geq 1$ and $row \geq 1$ do
    if $T_m(row, col)$ = 'diag' then
        Add $(row, col)$ into $L$.
        $col = col - 1$, $row = row - 1$.
    else if $T_m(row, col)$ = 'left' then
        $col = col - 1$.
    else
        $row = row - 1$.
    end if
end while
Return $Best\_Sum$, $L$
```

3.2 Subgradient descend

In order to find the tightest upper bound on $v(CMO)$ (or eventually to solve the problem), we need to solve in the dual space of the Lagrangian multipliers $LD = \min_{\lambda \geq 0} LR(\lambda)$, whereas $LR(\lambda)$ is a problem in x, y. A number of methods [30] have been proposed to solve Lagrangian duals : dual ascent, constraint generation, column generation, etc... Here, we choose the subgradient descent method [33], because of our large number of lagrangian multipliers. It is an iterative method in which at iteration t, given the current multiplier vector λ^t, a step is taken along a subgradient of $LR(\lambda)$; then, if necessary, the resulting point is projected onto the nonnegative orthant. It is well known that practical convergence of the subgradient method is unpredictable. For some problems, convergence is quick and fairly reliable, while other problems tend to produce erratic behavior of the multiplier sequence, or the Lagrangian value, or both. In a “good" case, one usually observes a sawtooth pattern in the Lagrangian values for the first iterations, followed by a roughly monotonic improvement and asymptotic convergence to a value that is hopefully the optimal Lagrangian bound. The computational runs on a rich set of real instances confirm a “good" case belonging to our approach at some expense in the speed of the convergence.

In our realization, the update scheme for λ_{ikj} (and analogously for λ_{ikl}) is $\lambda_{ikj}^{t+1} = \max\{0, \lambda_{ikj}^t - \Theta^t g_{ikj}^t\}$, where $g_{ikj}^t = \bar{x}_{ik} - \sum \bar{y}_{jlik}$ (see (2.9) and (2.10) for the sum definition) is the subgradient component $(0, 1$,or $-1)$, calculated on the optimal solution $\bar{x}$, $\bar{y}$ of $LR(\lambda^t)$. The step size Θ^t is $\Theta^t = \frac{\alpha(LR(\lambda^t)-Z_{lb})}{\sum(g_{ikj}^t)^2+\sum(g_{ikl}^t)^2}$ where Z_{lb} is a known lower bound for the CMO problem and α is a variable which is first initialized to 1, and then depends on the subgradient behavior. Every 5 consecutive improving subgradient iterations (i.e subgradient iteration which results either in a lower $LR(\lambda)$ or either in a bigger Z_{lb}), the α value is multiplied by 1.11, and every 5 consecutive non-improving subgradient iterations, it is divided by 1.11. In our experiments, this dynamic update of α proved to be more effective than the use of a fixed value.

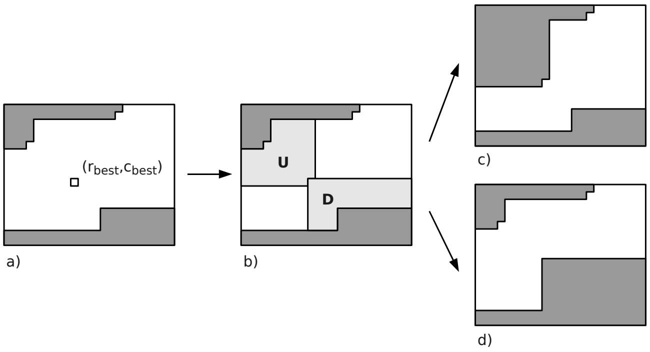
Into this approach the x-components of $LR(\lambda^t)$ solution provides a feasible solution to CMO and thus a lower bound also. The best one (incumbent) so far obtained is used for fathoming the nodes whose upper bound falls below the incumbent and also in section 5 for reporting the final gap. If $LD \leq v(CMO)$ then the problem is solved. If $LD > v(CMO)$ holds, in order to obtain the optimal solution, one could pass to a B&B algorithm suitably tailored for such an upper and lower bounds generator.

3.3 Branch and Bound

From among various possible nodes splitting rules, the one shown in **figure 2.6** gives good results (see section 5). Formally, a node of B&B is given by n_2 couples (b_k, t_k) for $k \in [1, n_2]$ which define the candidate vertex set $Cand$ (the white area in-between the two broken lines on figure 2.6). A vertex $j.l$ of the graph B' belongs to $Cand$ if $b_l \leq j \leq t_l$.

For any vertex $j.l$ in $Cand$, we can create two sets : $U(j,l) = \{i.k \in Cand$ such that $i.k \neq j.l, i \geq j$ and $k \leq l\}$ and $D(j,l) = \{i.k \in Cand$ such that $i \leq j$ and $k \geq l\}$. By definition, if an increasing subset of vertices has a vertex in $U(j,l)$, it cannot have one in $D(j,l)$, and vice versa.

Figure 2.6: Sketch of the B&B splitting strategy



a) The white area in-between broken lines represents the current node feasible set; b) Fixing the point *rbest.cbest* creates the two regions, D and U; c) and d) are the descendants of the node a).

Now, the two descendants of the current node are obtained by discarding from its feasible set the vertices belonging to the two respective domains $U(rbest, cbest)$ and $D(rbest, cbest)$ (see figure 2.6). The goal of this strategy is twofold: to create descendants that are balanced in sense of feasible set size and to reduce maximally the parent node's feasible set. Note that we experimentally found that applying this splitting rule in order to obtain four descendants (by dividing a problem in two sub-problems, and then by dividing each sub-problem in two) greatly speed up the branch and bound algorithm.

Finally, the main steps of the B&B algorithm are as follows:
Initialization: Set L={root} (root is the original CMO problem, i.e. with no restrictions on the set of vertices).
Problem selection and relaxation: Select and delete the problem
P from L having the biggest upper bound. Solve the Lagrangian dual of P.
Fathoming and Pruning: Follow classical rules.
Partitioning : Create and add to L the four descendants of P
Termination : if $L = \emptyset$, the solution (x^*, y^*) is optimal.

4 Relationship to other exact CMO approaches

Four other exact CMO approaches have been previously published : `B&Cut` [14], `Clique` [63], `LAGR` [11] and `CMOS` [69].

`B&Cut`, [14], was the first exact CMO solver. It has been outperformed by the recent exact algorithms, and we refer to its results only for the purpose of illustrating the progress in solving CMO.

`Clique` was designed by [63] who reformulated CMO as a maximum clique problem on a specially defined graph. The size of this graph (where any vertex corresponds to a common contact (overlap)) could be very large, and discovering the maximum clique in it is a difficult task.

`LAGR` was designed by [11] and similarly to our solver is based on Lagrangian relaxation. However, both algorithms differ in two major characteristics: (1) in the proposed MIP formulation ; and (2) in the set of dualized constraints. This can explain the significant discrepancies in the computational behavior. More precisely, in contrast to our formulation, the set of "increasing subsets of vertices" in [11] is defined by an exponential number of linear constraints (excluding the usage of any MIP solver); the variables y_{ikjl} are substituted by two variables y_{ikjl} and y_{jlik} and the dualized constraints are $y_{ikjl} = y_{jlik}$. This kind of Lagrangian relaxation falls in the class of the so called cost-split techniques.

`CMOS`, recently proposed by [69], is a direct CMO approach without using any mathematical programming models. The main result can be shortly derived in our Lagrangian relaxation approach as explained below. Consider both rectangles (right/left neighbors) associated to any vertex $i.k$ in Fig. 2.5. Denote by $p^+(i,k)/p^-(i,k)$ the longest feasible "path" with vertices in the right/left rectangle. Then $p^+(i,k)+p^-(i,k)$ is the length of the longest feasible "path" passing through $i.k$. Let us rewrite the objective function by using the substitutions $y_{ikjl} = 0.5(y_{ikjl}+y_{jlik})$ and let us add the constraint $y_{ikjl} = y_{jlik}$. The problem obtained by relaxing this constraint (equivalent to dualizing it with zero weight) is polynomialy solvable by DP algorithm. To check that solve the global problem from Theorem 3.1 by setting to $0.5(p^+(i,k) + p^-(i,k))$ the table entry at the position (i,k) (i.e. $T_g(i,k)$) and where $p^+(i,k)/p^-(i,k)$ is computed by solving the local problem in the rectangle $|\delta_1^+(i)| \times |\delta_2^+(k)|$ (respectively $|\delta_1^-(i)| \times |\delta_2^-(k)|$). The bound, say $U(CMOS)$, obtained in this manner is exactly the one given as Theorem 11 in [69]. Our initial bound $LR(0)$ in our subgradient descent is similar to $U(CMOS)$, but is obtained by using $p^+(i,k)$ instead of $0.5(p^+(i,k) + p^-(i,k))$. Our improved bound $LR(\lambda^*)$ is obtained at the expense of N subgradient iterations needed to solve the Lagrangian dual. We observed on the Skolnick set that $LR(\lambda^*)$ is on average about 25% smaller than $LR(0)$. We come to a principle question: is it worthwhile to use better, but N times more expensive bound instead of making zero subgradient iterations and to create N nodes in the B&B tree? To check this, we emulated `CMOS` with our algorithm and on all Skolnick instances the Lagrangian duality was definitely the winner.

5 Numerical results

The results presented here were obtained on a computer with an AMD Opteron CPU at 2.4 GHz and 4 Gb RAM. Our algorithm, denoted by `A_purva`[1], was implemented in C++. To generate contact maps we consider two amino-acids to be in contact if their C_α are within 7.5 Å, without taking into account contacts between consecutive amino-acids.

The protein benchmark sets we used are described in section 5.1. In section 5.2, we show that `A_purva` solves more CMO instances than other known exact CMO algorithms, and does it faster. In section 5.3, we show that even on instances that were not optimally solved, `A_purva`'s bounds are always tighter than the ones of `LAGR` (i.e. `A_purva`'s bounds are always closer to the optimum). Finally, in section 5.4, we successfully used `A_purva` to quickly obtain automatic classifications in very good agreement with the SCOP [4] ones.

[1] Apurva (Sanskrit) = not having existed before, unknown, wonderful, ...

Table 2.1: The Sokol set

Protein	Length	Species
1bpi	58	Cow (Bos taurus)
5pti	58	Cow (Bos taurus)
1knt	58	Human (Homo sapiens)
2knt	58	Human (Homo sapiens)
1era	62	Sea snake (Laticauda semifasciata)
3ebx	62	Sea snake (Laticauda semifasciata)
6ebx(A)	62	Sea snake (Laticauda semifasciata)

This set contain 7 small protein chains. The four first are from the “Small Kunitz-type inhibitors & BPTI-like toxins” SCOP family, and the last three are from the “Snake venom toxins” family.

Table 2.2: The five families in the Skolnick set.

	SCOP Family	Length	Proteins
1	CheY-related	120-130	1b00A, 1dbwA, 1natA, 1ntrA, 3chyA 1qmp(A,B,C,D), 4tmy(A,B)
2	Plastocyanin /azurin-like	97-105	1bawA, 1byo(A,B), 1kdiA, 1ninA 1plaA, 2b3iA, 2pcyA, 2pltA
3	Triosephosphate isomerase (TIM)	243-256	1amkA, 1aw2A, 1b9bA, 1btmA, 1htiA 1tmhA, 1treA, 1triA, 1ydvA, 3ypiA, 8timA
4	Ferritin	158-191	1b71A, 1bcfA, 1dpsA, 1fhaA, 1ierA, 1rcdA
5	Fungal ribonucleases	104	1rn1(A,B,C)

The Skolnick set contains 40 small protein chains from 33 proteins. They are classified by SCOP in five differant families.

5.1 Benchmark set descriptions

In the following experiments, we used three protein structure benchmark sets. The first one, known as Sokol set and described in **table 2.1**, was first introduced in [14] and was also used in [38,63,69]. It contains 7 small protein chains, whose number of amino-acids varies from 58 to 62. In the corresponding contact maps, the number of contacts varies from 177 to 197.

The second one, known as Skolnick set and described in **table 2.2**, was suggested by J. Skolnick and used in various recent papers related to protein structure comparison [11,38,54,69]. It contains 40 medium size chains / domains from 33 proteins. The number of amino-acids varies from 97 to 256, and in the corresponding contact maps, the number of contacts varies from 320 to 936. According to SCOP classification, the Skolnick set contains five families.

The last test set, denoted by Proteus_300, was proposed by us for the purpose of evaluating the capability of our algorithm to perform as a classifier. This is a large set containing more, and significantly longer proteins: 300 domains, with amino-acids number varying from 64 to 455. The maximum number of contacts is 1761. These domains are classified by SCOP in 24 folds, 27 super-families and 30 families.

For the interested reader, all our benchmarks and obtained results (solved instances, upper and lower bounds, run time, classifications...) are available on the URL:
http://www.irisa.fr/symbiose/softwares/resources/proteus300.

Table 2.3: Running time comparison on 10 Sokol set instances

			Running time (sec.)				A_purva
Instances	B&Cut[1]	Clique[2]	CMOS[3]	SADP[4]	LAGR[5]	A_purva[5]	sgd iter.
1bpi-1knt	331	19	0.27	0.05*	0.17	**0.01**	1
1bpi-2knt	423	182	0.46	0.06*	0.15	**0.01**	1
1bpi-5pti	320	30	0.00	0.05*	0.07	**0.01**	7
1knt-1bpi	331	110	0.43	0.06*	0.17	**0.01**	1
1knt-2knt	52	0	0.00	0.03	**0.01**	**0.01**	1
1knt-5pti	934	46	0.55	0.05*	0.13	**0.01**	5
2knt-5pti	760	95	0.37	0.05*	0.12	**0.01**	4
3ebx-1era	487	236	0.69	0.08	0.31	**0.01**	28
3ebx-6ebx	388	6	0.00	0.04*	**0.01**	**0.01**	1
6ebx-1era	427	101	0.38	0.04*	0.32	**0.01**	5

[1] Hardware similar to “Clique”. [2] SGI workstation $\sim$ 200MHz.
[3] Intel Pentium 4 $\sim$ 3GHz. * Instance not optimally solved.
[4] AMD Athlon64 3200+ $\sim$ 2GHz. [5] AMD Opteron $\sim$ 2.4GHz.

Running time in seconds of the different algorithms on 10 Sokol set instances. Note that only LAGR and A_purva were run on the same hardware. Other running times can only be used for order-of-magnitude comparisons. A_purva is clearly faster than the other algorithms, and is able to solve these small instances in less than 28 sub-gradient descent iterations (last column) on the root problem without branch and bound.
Note that for some instances, Clique and CMOS report running times of 0 seconds, which we believe correspond to running times smaller than 0.01 second. In our experiments (for LAGR and A_purva), these small values were rounded up to 0.01 second.

5.2 Performance

First, in **table 2.3**, we compare the time needed by the five exact algorithms – `B&Cut`, `Clique`, `LAGR`[2], `CMOS` and `A_purva` – plus a recent heuristic `SADP` [38]– for solving 10 instances from the Sokol set. Note that `B&Cut`, `Clique`, `CMOS` and `SADP` were run on different hardwares and contact maps. Thus, their running times are presented for order-of-magnitude comparison only. Data concerning `B&Cut` and `Clique` were taken from [63], the ones of `CMOS` from [69] and the ones of `SADP` from [38]. `A_purva` clearly outperforms all above mentioned algorithms. Note that many instances were solved by `A_purva` during the first iteration of sub-gradient descent (i.e. the optimal solution of $LR(\lambda_0)$ was also the optimal solution of $v(CMO)$). The other instances only needed few iteration of sub-gradient descent (see last column of table 2.3).

The second experiment consisted in using `LAGR`, `CMOS` and `A_purva` for aligning all 780 pairs of domains from the Skolnick Set. Again `LAGR` was executed on the same computer and with the same data as `A_purva`, while data concerning `CMOS` were taken from [69]. For all algorithms, the execution time was bound to 1800 seconds per instance[3]. As recapitulated in **table 2.4**, `A_purva` succeeded to solve 610 pairs, while `LAGR` and `CMOS` solved only 161 pairs.

We observed that the time for aligning similar structures (domains from the same family) varies between 0.02 sec. and 2.14 sec. (except for two instances). This time

[2] The code of LAGR was kindly provided to us by Giuseppe Lancia.

[3] This time limit, not mentioned in [69], was kindly provided to us by N. Sahinidis.

Table 2.4: Number of instances solved by the different CMO solvers

	LAGR[1]	A_purva[1]	CMOS[2]
Easy instances (164)	161	**164**	161
Hard instances (616)	0	**446**	0
Total (780)	161	**610**	161

[1] Run on AMD opteron ∼ 2.4GHz, with 7.5Å contact map.
[2] Run on Intel Pentium 4 ∼ 3GHz, with 7Å contact map.

Time limit was fixed to 1800 sec. per instance. Easy instances are the 164 pairs where both domains belong to the same SCOP family. Only A_purva is able to solve all easy instances, as well as some of the hard instances.

varies respectively from 3.47 sec. to more than 1800 sec. when aligning dissimilar structures (domains from different families). In this manner our results confirm once more the property (also observed in [11,69]) that : instances, such that both domains belong to the same family, seem to be easily solvable–in contrast to instances that align domains from different families. Note however, that this property is not shared with non CMO methods–exactly the opposite holds for FAST [71] where the average time for aligning similar structures is twice bigger then the one for aligning dissimilar ones. To the best of our knowledge, `A_purva` is the only solver able to solve many "hard" CMO instances (446 in the case of the Skolnick set).

Figure 2.7 compares the time needed by `LAGR` to the one of `A_purva` on the set of 161 Skolnick instances solved by both algorithms. We observe that `A_purva` is significantly faster than `LAGR`. More precisely, `LAGR` needed 10 h. 03 m. 30 seconds total time to solve these 161 instances while `A_purva` needed only 29.37 sec. Thus `A_purva` is about 1232 times faster than `LAGR` on this subset.

We also would like to mention that our approach seems to be not only the fastest exact CMO solver, but it is also noticeable faster than a recently published heuristic `VNS` [54] which solved the same 161 instances in 3 hs. 11 m. versus 29.37 sec. for `A_purva` (on similar workstations).

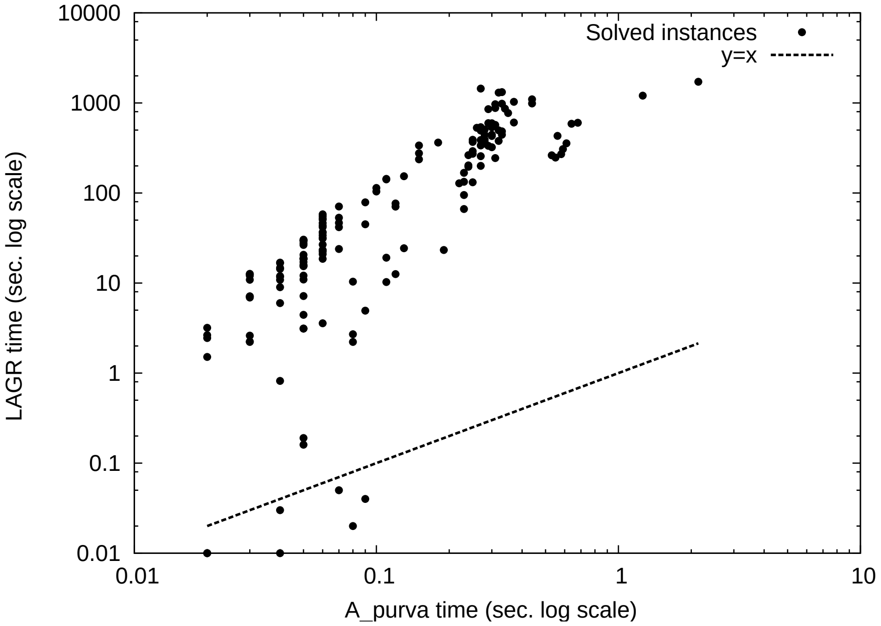
5.3 Quality of bounds

When a B&B type algorithm stops because of time limit (1800 sec. in our case), it provides an upperbound (UB) and a lowerbound (LB), which is a real advantage compared to any meta-heuristics. The relative gap value $\frac{UB-LB}{UB}$ measures how far is the optimization process from finding the exact optimum (small relative gap values relate to near optimality).

Our next observation concerns the quality of relative gaps obtained by `LAGR` and `A_purva` on the set of 170 Skolnick instances that both algorithms were not able to solve. **Figure 2.8** shows the relative gaps of `A_purva` plotted against those of `LAGR`. The entire is very asymmetric to the advantage of our algorithm since the relative gaps of `A_purva` are always smaller than those of `LAGR`, meaning that `A_purva` was always closer to the optimum.

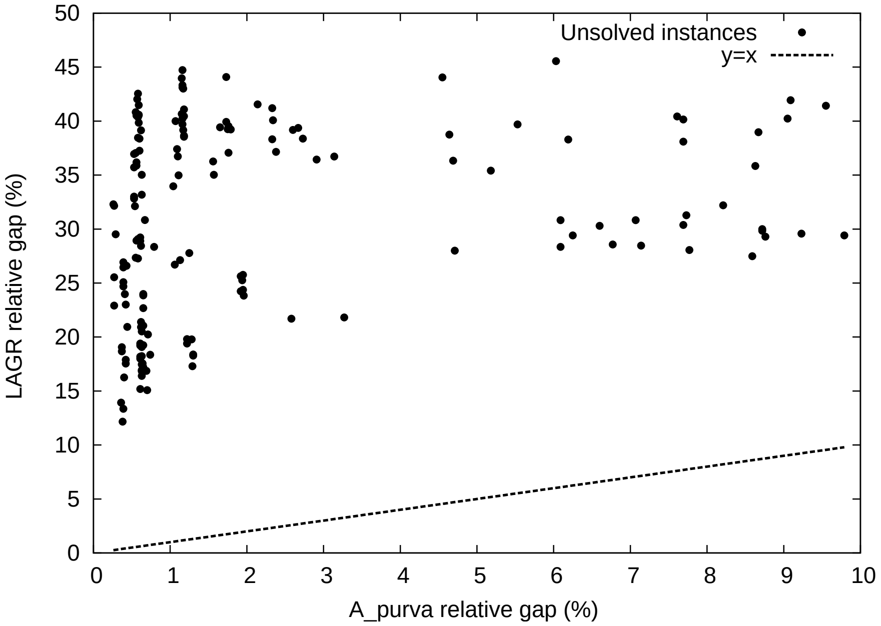
Even on the root of the branch and bound tree, our Lagrangian relaxation algorithm is still able to provide two bounds, an upperbound UB which is the smallest

Figure 2.7: A_purva versus LAGR running time comparison on the set of the 161 Skolnick instances solved by both algorithms



The A_purva time is presented on the x-axis, while the one of LAGR is on the y-axis. Often, A_purva is more than 1000 times faster.

Figure 2.8: Comparing relative gaps on the set of the 170 unsolved instances



The A_purva gaps (x-axis) are plotted against the LAGR gaps (y-axis). Any point is clearly above the line $y = x$ indicating that the relative gaps obtained by A_purva are substantially smaller.

Table 2.5: Five families chosen from Proteus_300 set for observing the relative duality gap divergences

	SCOP Family	SCOP Super-Family	SCOP Fold
1	Beta-glycanases	(Trans)glycosidases	TIM beta/alpha-barrel
2	Class I aldolase	Aldolase	TIM beta/alpha-barrel
3	AAT-like		PLP-dependent transferases
4	Extended AAA-ATPase domain	P-loop containing nucleoside triphosphate hydrolases	P-loop containing nucleoside triphosphate hydrolases
5	G proteins	P-loop containing nucleoside triphosphate hydrolases	P-loop containing nucleoside triphosphate hydrolases

Each family contains 10 instances. Families 1 and 2 come from the same fold but different super-families, family 3 is unique in its fold, while families 4 and 5 come from the same fold and same super-family.

$LR(\lambda)$ value found during the subgradient descent, and a lowerbound LB which is the incumbent value (the biggest value of $v(CMO)$ found so far). In this case, the relative gap value is called *Relative Duality Gap* (RDG).

In our results we observed that the RDG is smaller for instances in which both domains come from the same SCOP family. To illustrate this property, we extracted five families (presented in **table 2.5**) from Proteus_300 set. We run A_purva on this subset, using only 500 iterations of the sub-gradient descent on the root of branch & bound tree. **Table 2.6** presents the minimum, maximum and average value of the RDG for the corresponding instances. This gap equals zero (i.e. the instance was optimally solved) only for some of the pairs in which both domains come from the same family. Even if it is not zero, for such pairs the RDG is, in average, relatively small. Once again, such instances seem to be easily solvable, in contrast to instances for which the domains belong to different families. As we will see in the next section, this property (the smaller is the relative duality gap, the more similar are the domains) can be successfully used for classification purpose.

5.4 A_purva as a classifier

In this section we are interested in checking the ability of `A_purva` to perform as a classifier in a given small lapse of time. We used the following protocol. We limited the runs of `A_purva` to the root of the B&B tree, with a limit of 500 iterations for the subgradient descent. To evaluate the similarity between two proteins P_1 and P_2, we compared two measures. The first was the relative duality gap. To the best of our knowledge, this is the first attempt to use this function (considered as a dissimilarity - i.e. bigger duality gap relates to bigger dissimilarity) as a classifier. The second measure we have dealt with was the function[4] used in [69], where the similarity between two proteins P_1 and P_2 is given by $Sim(P_1, P_2) = \frac{2 \times LB}{|E_1|+|E_2|}$. The values computed by these two measures were given to `Chavl` [43, 44], a publicly available tool which proposes both a hierarchical ascendant classification and the cut corresponding to the best partition level (therefore, it does not require a similarity threshold). The obtained results were compared using the SCOP v1.73 classification

[4] this function is very close to the one used in [11]

Table 2.6: Relative duality gaps values

		1	2	3	4	5
	min	**0.000**	0.150	0.309	0.277	0.287
1	avg	**0.113**	0.298	0.401	0.354	0.412
	max	**0.305**	0.443	0.495	0.473	0.493
	min		**0.000**	0.278	0.200	0.178
2	avg		**0.114**	0.420	0.275	0.332
	max		**0.330**	0.545	0.366	0.460
	min			**0.000**	0.323	0.351
2	avg			$<$ **0.001**	0.421	0.458
	max			**0.016**	0.508	0.537
	min				**0.000**	0.184
4	avg				**0.005**	0.322
	max				**0.038**	0.429
	min					**0.000**
5	avg					$<$ **0.001**
	max					**0.003**

Minimum, average and maximum relative duality gaps obtained after 500 iterations of the subgradient descent concerning the five families from table 2.5. Pairs that belong to the same family are characterized by smaller relative duality gaps compared to pairs belonging to different families.

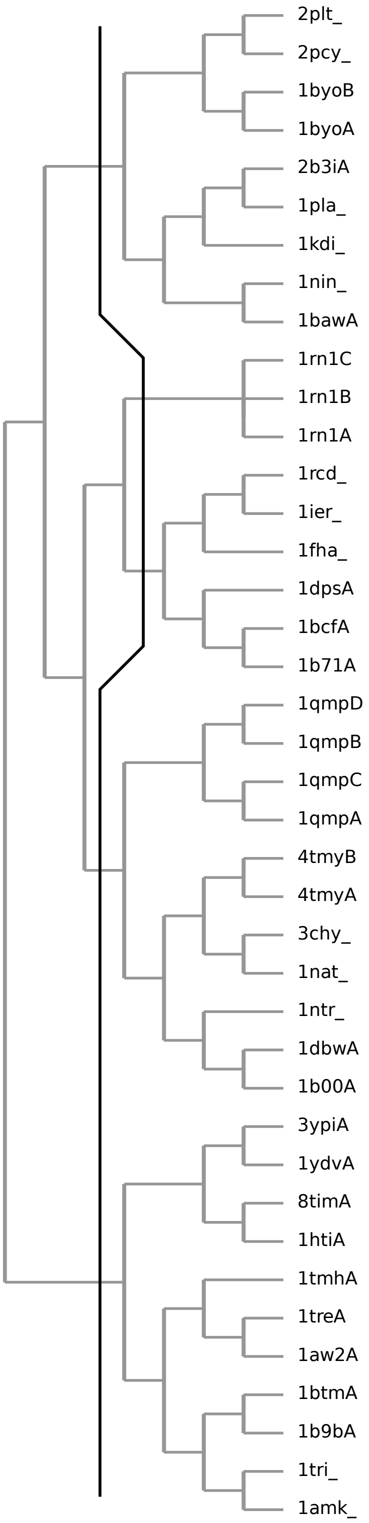
as a gold standard.

For the Skolnick set, the alignment of all pairs was done in less than 357 seconds ($\simeq$ 0.46 sec./pair). For both measures, the classification returned by `Chavl` was **exactly the same** as the classification at the family level in SCOP (which coincides with the fold level classification for the Skolnick set). The ascendant classification and the cut returned by Chavl when using the similarity function $Sim()$ is presented in **figure 2.9**. Note that in [38], `SADP` was also successfully used to automatically classify the Skolnick set, but required about 12 minutes on similar hardware.

To get a stronger confirmation of `A_purva` classifier capabilities, we performed the same operation on Proteus_300. Aligning all 44850 pairs required roughly 13h. 38m. ($\simeq$ 1.09 sec./pair).

In **table 2.7** we compare the SCOP classification at family level (1st column) with the classifications obtained using the relative duality gap and the similarity function $Sim(P_1, P_2)$. The relative duality gap classification (2nd column) contains 17 classes denoted by letters $A - Q$. Eleven of those classes correspond to SCOP families. The other classes are combinations of different SCOP families. This shows that either the relative duality gap is not specific enough for classification purpose (i.e. it will consider two proteins to be similar even if they are not), or more sophisticated tools are to be used. The results of the Sim function are given in the last column. The obtained classification contains 34 classes. The following four SCOP families: *L-arabinose binding protein-like*, *Tyrosine-dependent oxidoreductases*, *beta-glycanases* and *Class I aldolase* were each split in two in our classification. Such a divergence can be considered as a good result in the above context.

Figure 2.9: Ascendant classification of the Skolnick set.



Ascendant classification of the Skolnick set returned by Chavl when using the similarity function $Sim()$. Chavl also returns the cut corresponding to the best partition (in black).

Table 2.7: Comparing three classifications of Proteus_300 set

	Scop Families	RD-gap	*Sim*
1	WD40-repeat.	A	
2	N-acetyl transferase, NAT.	B	
3	C-type lectin domain.	C	
4	Cytochrome P450.	D	
5	Proteasome subunits.	E	
6	L-arabinose binding protein-like.	F	*
7	Phosphate binding protein-like.	G	
8	AAT-like.	H	
9	Protein kinases, catalytic subunit.	I	
10	Beta-glycanases.	J	*
11	Class I aldolase.	K	*
12	Ubiquitin-related.	L	
13	Enolase N-terminal domain-like.		
14	PDZ domain.		
15	DNA polymerase processivity factor.	M	
16	Lactate & malate dehydrogenases.		
17	HMA, heavy metal-associated domain.	N	
18	Canonical RBD.		
19	Fibronectin type III.		
20	C1 set domains (antibody constant domain-like).		
21	I set domains.		
22	Ferritin.	O	
23	Globins.		
24	Glutathione S-transferase (GST), C-terminal domain.		
25	LDH N-terminal domain-like.	P	
26	Tyrosine-dependent oxidoreductases.		*
27	G proteins.		
28	CheY-related.		
29	Nuclear receptor ligand-binding domain.	Q	
30	Extended AAA-ATPase domain.		

SCOP family classification of Proteus_300 set (1st column) versus the classifications obtained using the A_purva's relative duality gap (2nd column) and the *Sim* function (3rd column). The classification obtained by the *Sim* function is the same as the SCOP classification, except that four of the SCOP families (indicated by *) are each split in two into A_purva's classification.

5.5 Sensitivity and specificity analysis

In order to test the overall accuracy of `A_purva`, we checked its sensitivity and specificity with the SCOP classification v1.73 as the gold standard. For this purpose we have chosen from Proteus_300 four query structures (see **table 2.8**) with different level of difficulty for detecting their family affiliation. Note that:

- the family of d1t7ra_ is alone in its fold (an easy case);
- the family of d1uhva2 and the family *Class I aldolase* belong to two different super-families, but both are coming from the same fold (an intermediate case);
- the family of d1ny5a2 and the family *G proteins* belong to the same super-family and, hence, to the same fold (a hard case);
- detecting the affiliation of difp5a1 to its family is perturbed by the presence of two other families : the three families belong to the same fold, but only two of them come from the same super-family (the hardest considered case).

Each query belongs to a family of cardinality 10. For a fixed query we determined the sensitivities and specificities using 9 same-family pairs as the positive set and 290 different-family pairs as the negative set.

For comparison we also ran two other structure alignment algorithms (VAST [26] and Yakusa [13]) on the same dataset. The theoretical bases of these comparison methods differ significantly from CMO approach. Moreover, they belong to two distinct categories. VAST substantially uses the SSEs knowledge. It first supplies an exact SSEs alignment, and then, extends it heuristically (using a Gibbs sampling technique) to the amino-acids. In contrast to VAST, Yakusa directly establishes amino-acid-amino-acid correspondences. Its scoring scheme is based on the dihedral angles between the C_α atoms of four consecutive amino-acids. Both softwares are available to us[5] and we were able to manipulate easily large data sets on the local server. Both (VAST and Yakusa) are fast heuristics, in contrast to the exact CMO approach. For Proteus_300 (i.e. 44850 instances) the average running times were 0.02, 0.14, 2.25 sec/instance for Yakusa, VAST and `A_purva` respectively.

Table 2.9 summarizes the detected specificity at fixed sensitivity cutoffs, while **table 2.10** summarizes the detected sensitivity at fixed specificity cutoffs. Globally, we observe that `A_purva` is more accurate than VAST, which is more accurate than Yakusa. Even more, tables 2.9 and 2.10 show that `A_purva` was able to detect all pairs of structures designated by SCOP to be in the same family without errors, and this - for all four queries. Hence, on the observed dataset `A_purva` achieves the highest specificity even at the highest sensitivity, and vice versa.

6 Conclusion

In this chapter, we give an efficient exact algorithm for contact map overlap problem. The bounds are found by using Lagrangian relaxation, and the dual problem is solved by sub-gradient approach. The performance of the algorithm is demonstrated on

[5] Many thanks to Jean-François Gibrat and Joel Pothier for kindly providing us the source code of VAST and Yakusa respectively.

Table 2.8: The four queries and their neighbourhoud in the Proteus_300 set.

Query	Family	Super-Family	Fold
d1t7ra_	Nuclear receptor ligand-binding domain	Nuclear receptor ligand-binding domain	Nuclear receptor ligand-binding domain
d1uhva2	beta-glycanases	(Trans)glycosidases	TIM beta/alpha-barrel
	Class I aldolase	Aldolase	TIM beta/alpha-barrel
d1ny5a2	Extended AAA-ATPase domain	P-loop containing nucleoside triphosphate hydrolases	P-loop containing nucleoside triphosphate hydrolases
	G proteins	P-loop containing nucleoside triphosphate hydrolases	P-loop containing nucleoside triphosphate hydrolases
difp5a1	C1 set domains (antibody constant domain-like)	Immunoglobulin	Immunoglobulin-like beta-sandwich
	I set domains	Immunoglobulin	Immunoglobulin-like beta-sandwich
	Fibronectin type III	Fibronectin type III	Immunoglobulin-like beta-sandwich

Four queries (1^{st} column) and their SCOP classification.

the Skolnick set and its superiority over the existing algorithms is obvious. The capability of the proposed algorithm to provide a similarity measure was tested on a large data set of 300 protein domains. We were able to obtain in a short time a classification in very good agreement to the well known SCOP database.

Table 2.9: Specificity at three sensitivity cutoffs.

Query	Sensitivity (number of corresponding true positives)	Specificity (number of corresponding false positive)		
		A_purva	VAST	Yakusa
d1t7ra_	100% (9)	100% (0)	100% (0)	99.7% (1)
	88.9% (8)	100% (0)	100% (0)	100% (0)
	77.8% (7)	100% (0)	100% (0)	100% (0)
d1uhva2	100% (9)	100% (0)	97.2% (8)	91.7% (24)
	88.9% (8)	100% (0)	99.3% (2)	99.7% (1)
	77.8% (7)	100% (0)	100% (0)	100% (0)
d1ny5a2	100% (9)	100% (0)	100% (0)	60.7% (114)
	88.9% (8)	100% (0)	100% (0)	90.7% (27)
	77.8% (7)	100% (0)	100% (0)	97.9% (6)
d1fp5a1	100% (9)	100% (0)	98.6% (4)	25.5% (216)
	88.9% (8)	100% (0)	99.7% (1)	93.1% (20)
	77.8% (7)	100% (0)	100% (0)	95.2% (14)

Specificity at three sensitivity cutoffs computed for the queries from table 2.8. True positives are defined as domains coming from the scop family of the query.

Table 2.10: Sensitivity at three specificity cutoffs.

Query	Specificity (number of corresponding false positive)	Sensitivity (number of corresponding true positives)		
		A_purva	VAST	Yakusa
d1t7ra_	100% (0)	100% (9)	100% (9)	88.9% (8)
	99.6% (1)	100% (9)	100% (9)	100% (9)
	98.3% (5)	100% (9)	100% (9)	100% (9)
d1uhva2	100% (0)	100% (9)	77.8% (7)	77.8% (7)
	99.6% (1)	100% (9)	77.8% (7)	88.9% (8)
	98.3% (5)	100% (9)	88.89% (8)	88.9% (8)
d1ny5a2	100% (0)	100% (9)	100% (9)	44.4% (4)
	99.6% (1)	100% (9)	100% (9)	44.4% (4)
	98.3% (5)	100% (9)	100% (9)	66.7% (6)
d1fp5a1	100% (0)	100% (9)	77.8% (7)	11.1% (1)
	99.6% (1)	100% (9)	88.9% (8)	11.1% (1)
	98.3% (5)	100% (9)	100% (9)	11.1% (1)

Sensitivity at three specificity cutoffs computed for the queries from table 2.8. False positives are defined as domains which does not come from the scop family of the query.

Chapter 3

Speeding up CMO by using biological knowledge

1 Problematics

In the previous chapter, we showed that the contact map overlap maximization is an efficient scoring scheme, in the sense that the scores it associates to the pairs of protein structures allow good automatic classifications. However, the pairwise comparisons are still too much time consuming for using CMO in the context of large database classification.

Since the CMO problem is NP-Hard, a good approach for speeding-up the solving process is to reduce the size of the input data, which in our case is the alignment graph. In a first step, we were inspired by structural biology to use the Secondary Structure Elements (SSE) of the proteins for creating vertex-filters for the alignment graph. In a second step, to further exploit the secondary structure information, we define a hierarchical approach for solving the CMO problem. Each time, the goal is twofold. On the one hand, the alignment graph gets sparser, and thus the solving process gets faster. On the other hand, the obtained alignment should be more biologically acceptable.

The ideas presented in this chapter are presented for the first time in the context of CMO. We are not aware of any attempt to include secondary structure knowledge into CMO, thus it was mentioned as a possible future work in [69]. Hierarchical approaches have already been used in the context of protein structure comparison, for example in VAST [26] or in CATHEDRAL [57], but no hierarchical approach was proposed for CMO.

2 Secondary structure based filters

Reducing an alignment graph $G = (V, E)$ can be done either at the vertex level, by removing vertices $i.k$ from V according to some rules saying that amino-acid i from P_1 is not compatible with amino-acid k from P_2, or at the edge level, by removing edges $(i.k, j.l)$ from E according to some rules saying that matching i from P_1 with amino-acid k from P_2 is not compatible with matching j from P_1 with amino-acid l from P_2. Note that removing a vertex $i.k$ also removes the edges connected to it, so it seems to be a good idea to start reducing the alignment graph by removing vertices.

As stated in the thesis introduction, the protein structure can be described at different levels : at the primary structure level by using its sequence of amino acids, at the secondary structure level by using its secondary structure elements, or at the tertiary/quaternary structure levels by using the 3D coordinates of its amino-acids atoms. In this chapter, we decided to use the secondary structure knowledge for designing our filters, because most molecular biologists will agree that an α-helix should not be matched with a β-strand, while using sequence information (like the physical properties of amino-acids) or tertiary/quaternary structure informations (like angles between amino-acids) is more delicate.

In its original definition, CMO deals with a simplification of the three dimensional structure of proteins which is the notion of contact between amino-acids, and does not consider secondary structure informations. Thus, any amino-acid coming from an α-helix can be matched with an amino-acid coming from a β-strand if this matching contributes positively to the objective function. Potentially, this conducts to biologically unacceptable matching. Based on theses observations, we designed two filters which prohibit such undesirable matchings by removing the corresponding vertices from the alignment graph.

SSE filter 1: The goal of the first SSE filter is to prohibit matchings between amino-acids from α-helices and amino-acids from β-strands. Thanks to secondary structure assignment software like DSSP [39], Kaksi [45] or Stride [20], it is possible to create a function $Type(P, i)$ which associates to the i^{th} amino-acid of a protein P the kind of secondary structure to which it belongs.

$$Type(P,i) = \begin{cases} \alpha & \text{if the } i^{th} \text{ amino-acid of protein } P \text{ belongs to an } \alpha\text{-helix} \\ \beta & \text{if it belongs to a } \beta\text{-strand} \\ . & \text{if it does not belong to a SSE (i.e. it belongs to a loop).} \end{cases} \tag{3.1}$$

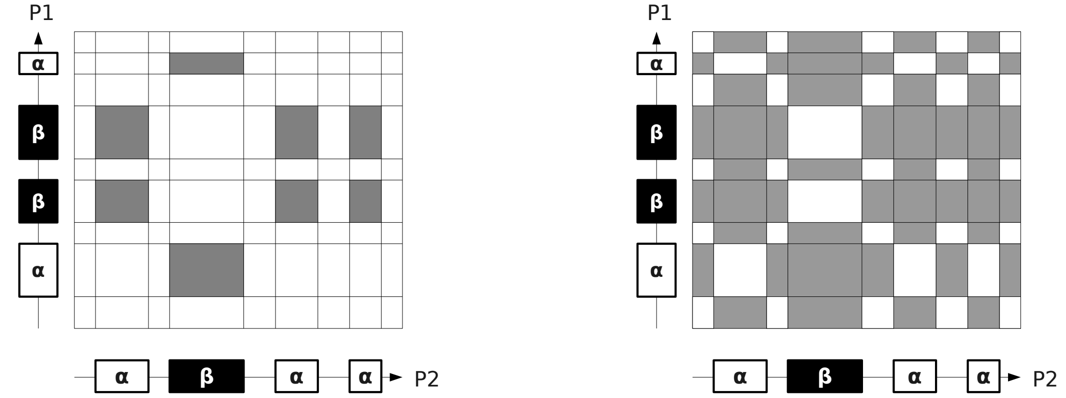
Given an alignment graph $G = (V, E)$ and a secondary structure assignment function $Type()$, prohibiting matchings between α-helices and β-strands is done by removing from G the vertices $i.k \in V$ (and the corresponding edges) such that (1) $Type(P_1, i) = \alpha$ and $Type(P_2, k) = \beta$, or (ii) $Type(P_1, i) = \beta$ and $Type(P_2, k) = \alpha$.

SSE filter 2: The first SSE filter still allows matching between a secondary structure (α-helix or β-strand) and a loop. The goal of the second SSE filter is to also prohibit such matchings. This is done by removing from the alignment graph $G = (V, E)$ the vertices $i.k \in V$ (and the corresponding edges) such that $Type(P_1, i)$ $\neq Type(P_2, k)$.

The effects of these two SSE filters on the alignment graph are illustrated in **figure 3.1**. Note that an optimal solution found on a filtered graph (by using SSE filter 1 or 2) is a feasible solution of the original problem, but is not necessarily an optimal solution of the original problem.

As shown in the result section, using the SSE filters greatly reduces the computational time of A_purva. This led us to further exploit the secondary structure knowledge for solving the contact map overlap.

Figure 3.1: Effect of the SSE filters on the alignment graph.



In the original CMO, all amino-acids from P1 can be matched with all amino-acids from P2. **Left:** SSE filter 1 prohibits matching between amino-acids coming from an α-helix and amino-acids coming from a β-strand (prohibited matchings are represented by grey areas on the alignment graph). **Right:** SSE filter 2 also prohibits matching between amino-acids coming from a secondary structure (α-helix or β-strand) with amino-acids coming from a loop.

3 The hierarchical approach

We were inspired by VAST for better using the secondary structure information. VAST uses the following hierarchical approach : first, a “low resolution” alignment is done at the secondary structure level, and second, this secondary structure alignment is used to filter the “high resolution” alignment at the amino-acids level. This strategy is interesting because the number of secondary structure element in a protein is much smaller than the number of amino-acid (as illustrated in **table 3.1** on the Skolnick set), and thus the time spend for solving the secondary structure alignment is more than justified by the gain when solving the amino-acid alignment. In this section, we aim at producing a similar hierarchical approach for CMO, that is, to first align secondary structures, and then to used this alignment for filtering the amino-acids alignment graph.

3.1 SSEs alignment, the contact map approach

In the literature, the two main references for aligning secondary structure of protein are : (i) VAST, the protein structure comparison software used in the NCBI for determining the neighbourhood of a protein structure in the ENTREZ database[1], and (ii) GRATH [31], the secondary structure alignment algorithm used in the structural classification of protein CATH, more precisely in the domain boundary recognition algorithm CATHEDRAL.

Using VAST to align SSE and then A_purva for the amino-acids has been implemented, but not really tested. The main reason is practical: VAST only returns secondary structure alignment for similar proteins, when we are interested in the opposite cases. In effect, A_purva easily solves the CMO problems corresponding

[1] www.ncbi.nlm.nih.gov/Entrez

Table 3.1: Amino-acid and secondary structure element number comparison in the Skolnick set.

SCOP Family	N^o of AA	N^o of SSEs	Protein chains
CheY-related	120-130	9-10	1b00A, 1dbwA, 1natA, 1ntrA, 3chyA 1qmp(A,B,C,D), 4tmy(A,B)
Plastocyanin /azurin-like	97-105	7-10	1bawA, 1byo(A,B), 1kdiA, 1ninA 1plaA, 2b3iA, 2pcyA, 2pltA
Triosephosphate isomerase (TIM)	243-256	19-20	1amkA, 1aw2A, 1b9bA, 1btmA, 1htiA 1tmhA, 1treA, 1triA, 1ydvA, 3ypiA, 8timA
Ferritin	158-191	5-8	1b71A, 1bcfA, 1dpsA, 1fhaA, 1ierA, 1rcdA
Fungal ribonucleases	104	6-8	1rn1(A,B,C)

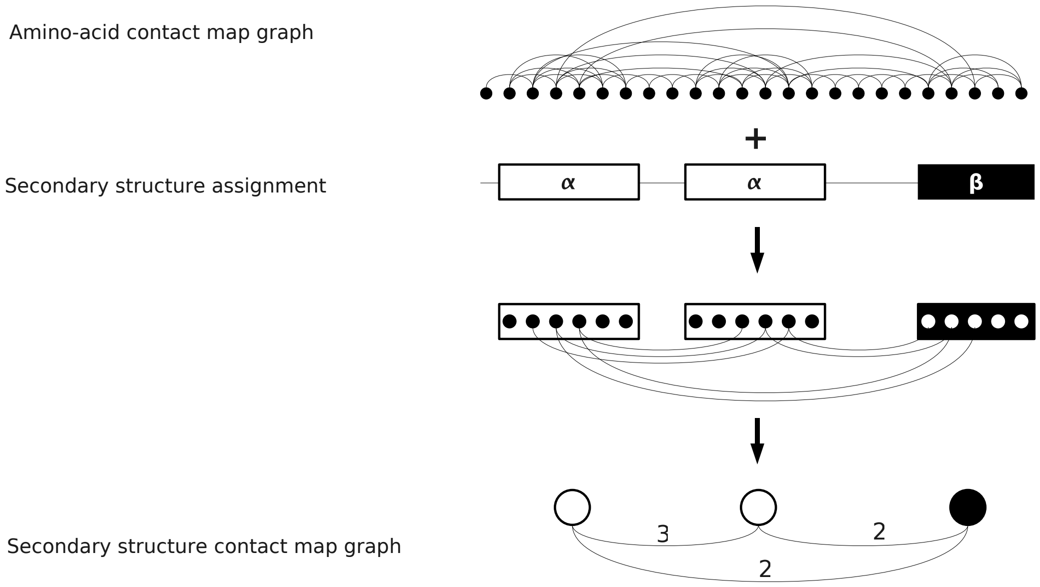
For each family of the Skolnick set, the number of amino-acids in the protein chains (presented in the second column) is compared to the number of secondary structure elements (presented in the third column). For example, in the CheY-related family, the number of secondary structure element is about 13 times smaller than the number of amino-acids.

to similar structures, while it is not always able to solve (within a reasonable amount of time) the CMO problems corresponding to dissimilar structures. Moreover, we wanted to stay in the context of CMO. This motivated us to propose a novel approach for aligning secondary structure elements based on contact maps. To the best of our knowledge, there is no such secondary structure alignment method based on CMO, and thus we needed to both define how to create and how to align secondary structure contact maps.

Secondary structure contact maps

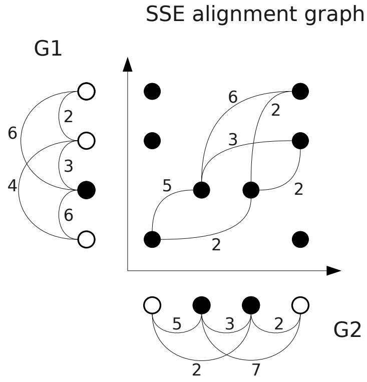
The secondary structure contact map graph of a protein is a derivative of the amino acid contact map graph. In a first step, an amino-acid contact map graph is created by using the same protocol as in chapter one (by using a distance threshold of 7.5 Å between α-carbons, and without taking into account contacts between successive amino-acids). The composition of the protein in terms of secondary structure is determined by using a secondary structure assignment software (we used Kaksi [45], though any other secondary structure assignment tool can be used). Note that the secondary structure elements are also ordered according to their positions in the protein chain. The secondary structure contact map of a protein P is a weighted graph $G = (V, E, W)$ in which each vertex of the ordered vertex set V correspond to a secondary structure element of P (the i^{th} vertex of V corresponds to the i^{th} secondary structure element of P). To each vertex is also associated the type of its corresponding secondary structure element (either α or β). A contact edge (i, j) between vertices i and j exists if and only if there exist at least one contact between the amino-acids from SSE i and the amino-acids from SSE j. To each edge (i, j) is associated a weight $w_{ij} \in W$ which is equal to the number of contacts between the amino-acids from SSE i and the amino-acids from SSE j. This process is illustrated in **figure 3.2**.

Figure 3.2: Secondary structure contact map graph creation.


Amino-acid contact map graph
+
Secondary structure assignment
α
α
β
Secondary structure contact map graph
3
2
2

Each vertex of the secondary structure contact map correspond to a secondary structure element, and to each vertex is associated the type of the corresponding secondary structure element which is either α-helix (in white) or β-strand (in black). Two secondary structure element are connected by a contact edge if there is at least one contact between their amino-acids. To each contact edge is associated a value which represents the exact number of amino-acid contacts between the secondary structure elements.

Figure 3.3: Secondary structure alignment graph.



The secondary structure alignment of proteins P_1 and P_2 (represented by their secondary structure contact map graphs $G_1 = (V_1, E_1, W_1)$ and $G_2 = (V_2, E_2, W_2)$) is modelled in a $|V_1| \times |V_2|$ alignment graph $G = (V, E, W)$. Vertex $i.k$ is in V if $i \in V_1$ and $k \in V_2$ are from the same type (α-helix, in white, or β-strand, in black). Edge $(i.k, j.l) \in E$ if and only if (i) $i < j$ and $k < l$, and (2) if edge $(i.j) \in E_1$ and edge $(k.l) \in E_2$. To each edge $(i.k, j.l) \in E$ is associated a weight $w_{ikjl} \in W$ which is equal to $\min(w1_{ij}, w2_{kl})$.

Secondary structure alignment

Aligning two secondary structure contact maps is similar to aligning two amino-acid contact maps. Given two proteins P_1 and P_2, and their corresponding secondary structure contact map graphs $G1 = (V_1, E_1, W_1)$ and $G_2 = (V_2, E_2, W_2)$, we are looking for an order preserving matching between element of V_1 and V_2. Matching $i \leftrightarrow k$ is possible only if i and k are compatible, i.e. if both are α-helices or if both are β-strands. The contribution of matching $i \leftrightarrow k$ and $j \leftrightarrow l$ is $w_{ikjl} = min(w1_{ij}, w2_{kl})$, as this value represents the maximum number of common contacts between the amino-acids induced by the secondary structure matching.

As illustrated in **figure 3.3**, the secondary structure alignment is then modelled in an $|V_1| \times |V_2|$ alignment graph $G = (V, E, W)$. A vertex $i.k$ exists (i.e. $i.k \in V$) if SSE $i \in V_1$ and SSE $k \in V_2$ are from the same type. An edge $(i.k, j.l)$ exists (i.e. $(i.k, j.l) \in E$) if and only if (i) $i < j$ and $k < l$, for order preserving, and (2) if edge $(i.j) \in E_1$ and edge $(k.l) \in E_2$. To each edge $(i.k, j.l) \in E$ is associated a weight $w_{ikjl} \in W$ which is equal to $\min(w1_{ij}, w2_{kl})$. Like in the original CMO, we are looking in G for the maximum edge-weighted increasing subset of vertices, the only difference being that the weights associated to the edges are not necessarily equal to 1, but are in $\mathbb{N} - \{0\}$.

Thus, the integer programming model for solving the weighted CMO problem is similar to the one for solving the unweighed CMO problem, except for the objective function.

We associate to each vertex $i.k \in V$ a binary variable x_{ik} such that :

$$x_{ik} = \begin{cases} 1 & \text{if vertex } i.k \text{ is in the increasing subset of vertices,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

And we associate to each edge $(i.k, j.l) \in E$ a binary variable y_{ikjl} such that :

$$y_{ikjl} = \begin{cases} 1 & \text{if edge } (i.k, j.l) \text{ is in the increasing subset of vertices,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The new objective function is to find the maximum sum of edge weights :

$$v(Weighted\_CMO) = \max \sum_{(i.k,j.l) \in E'} w_{ikjl} \times y_{ikjl} \tag{3.4}$$

Subject to :

$$x_{ik} \geq \sum_{l \in \delta_2^+(k)} y_{ikjl}, \qquad j \in \delta_1^+(i), \quad i \in [1, n_1 - 1], \quad k \in [1, n_2 - 1]. \tag{3.5}$$

$$x_{ik} \geq \sum_{j \in \delta_1^+(i)} y_{ikjl}, \qquad l \in \delta_2^+(k), \quad i \in [1, n_1 - 1], \quad k \in [1, n_2 - 1]. \tag{3.6}$$

$$x_{ik} \geq \sum_{(r,s) \in row_{ik}(j)} y_{rsik}, \qquad j \in \delta_1^-(i), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{3.7}$$

$$x_{ik} \geq \sum_{(r,s) \in col_{ik}(l)} y_{rsik}, \qquad l \in \delta_2^-(k), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{3.8}$$

$$\sum_{l=1}^{k} x_{il} + \sum_{j=1}^{i-1} x_{jk} \leq 1, \qquad i \in [1, n_1], \quad k \in [1, n_2]. \tag{3.9}$$

$$\sum_{l \in \delta_2^+(b), l \leq k} y_{abil} + \sum_{j \in \delta_1^+(a), j < i} y_{abjl} \leq 1, a \in [1, n_1], b \in [1, n_2], i \in \delta_1^+(a), k \in \delta_2^+(b) \tag{3.10}$$

Our CMO solver, A_purva, was modified accordingly.

3.2 Extension to the amino-acids

The principle is to extend the alignment obtained at the secondary structure level to the amino-acid. Being given two proteins P_1 and P_2, a secondary structure alignment "$i_1 \leftrightarrow k_1$, ..., $i_n \leftrightarrow k_n$" and an amino-acid alignment graph $G = (V, E)$, this is done by removing from the alignment graph all vertices i.k which do not satify at least one of the following conditions:

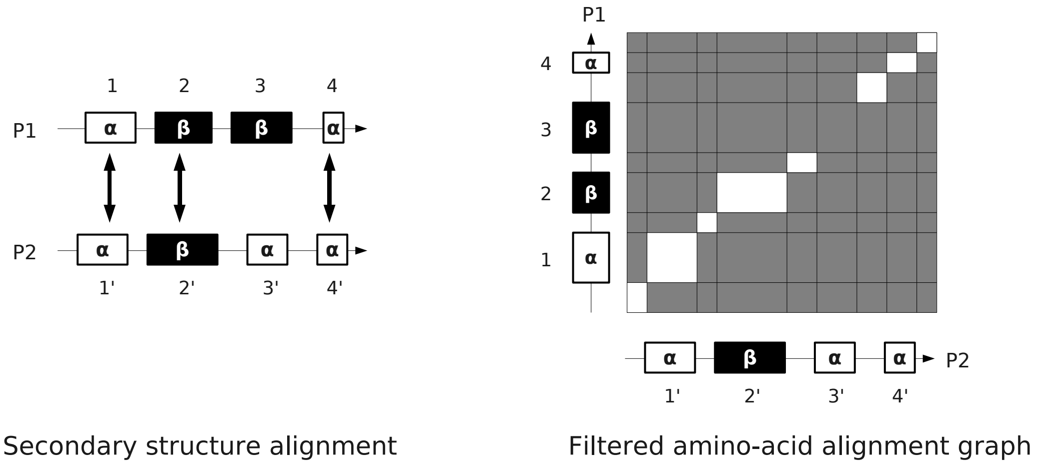
1. Amino-acid i comes from SSE i_j, amino-acid k comes from SSE k_j, and $i_j \leftrightarrow k_j$ is in the secondary structure alignment.

2. Amino-acid i comes from the loop before SSE i_j, amino-acid k comes from the loop before SSE k_j, and $i_j \leftrightarrow k_j$ is in the secondary structure alignment.

3. Amino-acid i comes from the loop after SSE i_j, amino-acid k comes from the loop after SSE k_j, and $i_j \leftrightarrow k_j$ is in the secondary structure alignment.

The effect of this filter is illustrated in **figure 3.4**.

Figure 3.4: Illustration of the hierarchical filter.



Secondary structure alignment

Filtered amino-acid alignment graph

Left : Secondary structure alignment “$1 \leftrightarrow 1', 2 \leftrightarrow 2', 4 \leftrightarrow 4'$”. Right : The corresponding filtered amino-acids alignment graph, when applying the rules discribed in section 3.2.

4 Results

The results presented in this chapter were all obtained on the GenOuest bioinformatics platform, “GenoCluster2”, a cluster of computers with Intel Xeon E5462 processor at 2.8 GHz and 32GB of memory, running under Red Hat Enterprise Linux Server release 5.1. Our CMO solver, A_purva, is compared to its filtered counterparts, which are denoted as follows : When the alignment graph is filtered by using SSE filter 1, the solver is denoted by A_purva_SSE1. When the alignment graph is filtered by using SSE filter 2, the solver is denoted by A_purva_SSE2. When CMO is solved by using the hierarchical approach, the solver is denoted by A_purva_H. The comparison of all these approaches are done on the Skolnick set (previously presented in table 3.1), and the corresponding contact map graphs were generated by using a distance threshold of 7.5 Å between the α-carbons, without taking into account the contacts between consecutive amino-acids.

In section 4.1 we compare A_purva to its filtered counterparts A_purva_SSE1 and A_purva_SSE2 to illustrate the effect of the secondary structure based filters. Then, in section 4.2, we present the secondary structure alignment obtained by the first stage of A_purva_H. Finally, in section 4.3 we compare A_purva to its hierarchical counterpart A_purva_H.

4.1 Effect of using SSE filters

In this section, we show that using SSE filter 1 and SSE filter 2 greatly reduces the running times of A_purva. We also show that even on filtered alignment graphs, the scores returned by A_purva_SSE1 and A_purva_SSE2 still allow good automatic classifications of protein structures.

Solved instances comparison

The first experiment consisted in aligning all 780 pairs of domains from the Skolnick set with A_purva (when alignment graphs are not filtered), A_purva_SSE1 (when alignment graphs are filtered by SSE filter 1) and A_purva_SSE2 (when alignment graphs are filtered by SSE filter 2). In all cases, execution time was bounded to 1800 seconds per instance.

Table 3.2 presents the number of instance solved by each algorithm. Using SSE filter 1 allows A_purva to solve 25 more instances, while using SSE filter 2 allows A_purva to solve 138 more instances

Table 3.2: Effects of SSE filters on the number of solved instances.

	A_purva (without filter)	A_purva_SSE1 (with SSE filter 1)	A_purva_SSE2 (with SSE filter 2)
Similar instances (164)	**164**	**164**	**164**
Dissimilar instances (616)	465	492	**603**
Total (780)	629	654	**767**

Time limit was fixed to 1800 sec. per instance. Similar instances are the 164 pairs where both domains belong to the same SCOP family. Using SSE filter 1 allows A_purva to solve 25 more instances than without using it, while using SSE filter 2 allows to solve 138 more instances than without filtering.

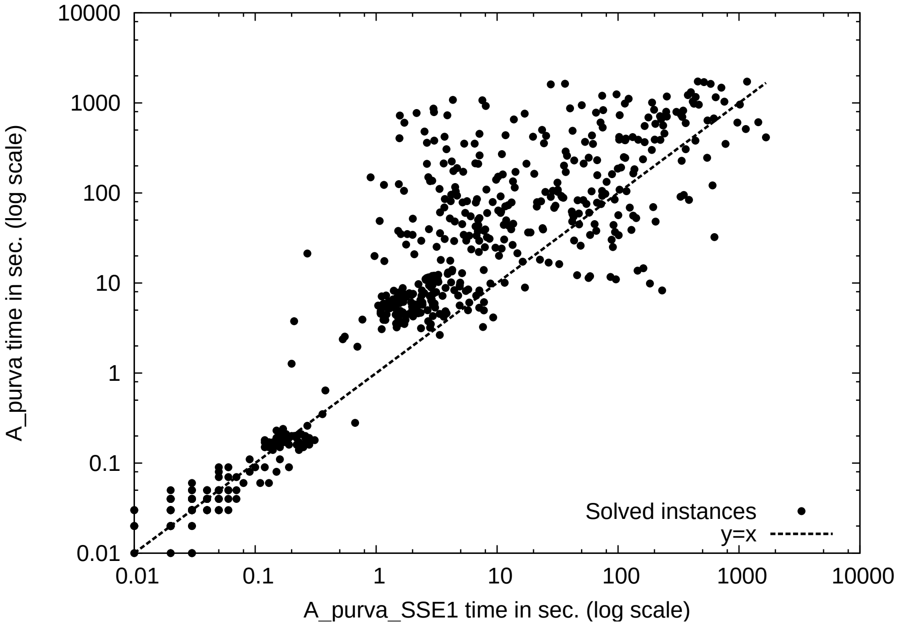
Amongst the 584 Skolnick instances which are solved by both A_purva and A_purva_SSE1, we observed that the optimal solution found by A_purva_SSE1 was equal to the one of A_purva in 266 cases. However, this only happens once between the 619 Skolnick instances which are solved by both A_purva and A_purva-_SSE2.

Figure 3.5 compares the running time of A_purva to the one of A_purva_SSE1 on the set of 584 Skolnick instances solved by both algorithms. On average, using SSE filter 1 makes A_purva 2.53 times faster (up to 461.48 times). **Figure 3.6** does the same comparison between A_purva and A_purva_SSE2, on the set of 619 Skolnick instances solved by both algorithms. On average, using SSE filter 2 makes A_purva 48.69 times faster (up to 15041.13). These results were expected, since SSE filter 2 removes much more vertices from the alignment graphs than SSE filter 1. However, note that the SSE filters are efficient only for instances which required more than one second of computational time.

Automatic classifications

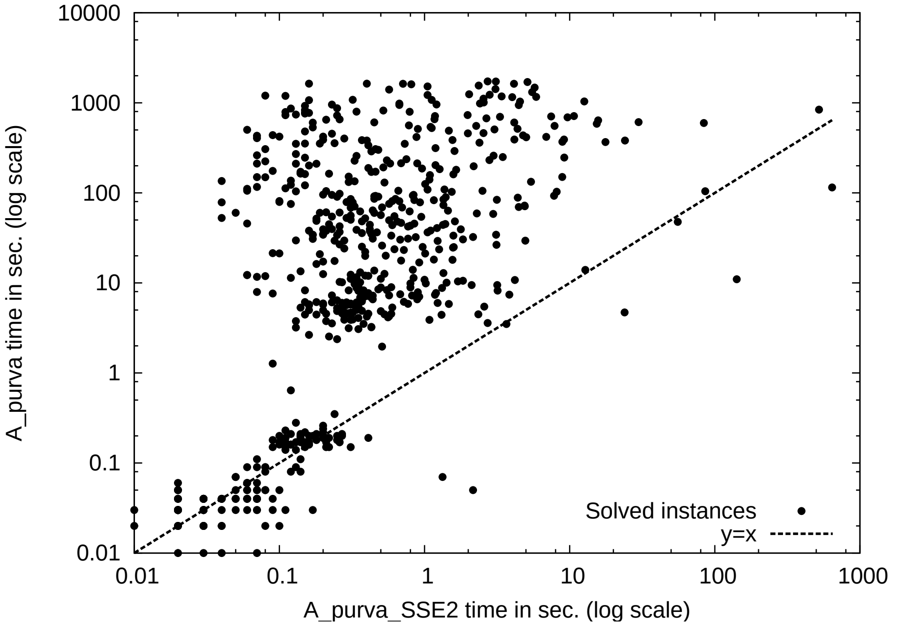
The second experiment consists in testing the ability of A_purva_SSE1 and of A_purva_SSE2 to quickly provide good scores for automatic classifications. The runs of both methods were limited to the root problems (i.e. without branch and bound) and to 500 sub-gradient iterations. We used the score defined in [69] which is $Sim(P_1, P_2) = \frac{2 \times LB}{|E_1|+|E_2|}$, where LB denotes the number of common contacts in the best feasible solution found by the algorithm, and where E_1 and E_2 denote the number of edges in the contact map graphs of P_1 and P_2. These scores, computed for the Skolnick set, were given to Chavl, and the corresponding classifications are

Figure 3.5: A_purva and A_purva_SSE1 running time comparison.



For each of the 584 Skolnick instances solved by both methods, the running time of A_purva_SSE1 (x-axis) is plotted against the running time of A_purva (y-axis). In average, A_purva_SSE1 is 2.53 times faster than A_purva, and this difference goes up to 461.48 times.

Figure 3.6: A_purva and A_purva_SSE2 running time comparison.



For each of the 619 Skolnick instances solved by both methods, the running time of A_purva_SSE2 (x-axis) is plotted against the running time of A_purva (y-axis). In average, A_purva_SSE2 is 48.69 times faster than A_purva, and this difference goes up to 15041.13 times.

compared to the family level of the SCOP classification (v1.73) used as a gold standard.

Figure 3.7 presents the ascendant classification and the corresponding best partitioning cut returned by Chavl when using the scores computed by A_purva_SSE1. The obtained classification is identical to the SCOP's one at family level, and was obtained in 365.98 seconds (0.47 second per instance in average). This is 1.23 times slower than the 297.67 sec needed by A_purva to obtain the same results.

Figure 3.8 presents the ascendant classification and the corresponding best partitioning cut returned by Chavl when using the scores computed by A_purva_SSE2. The obtained classification is identical to the SCOP's one at family level, and was obtained in 201.21 seconds (0.26 second per instance in average), which is about 1.48 times faster than when using A_purva.

4.2 Secondary structure alignment

First, we tested the ability of A_purva to solve the secondary structure alignment instances from the Skolnick set. We created the secondary structure contact map by using a 7.5 Å distance threshold between α-carbons, and by using the secondary structure assignment returned by Kaksi. Since secondary structure instances are much smaller than the amino-acid instances, we set the time limit per instance to 1 second. **Table 3.3** shows that A_purva was able to solve 777 secondary structure instances over 780 within such time limit. Moreover, for these 777 instances, the average running time was less than 0.01 sec. (up to 0.05 sec.). For the three unsolved instance, the absolute gap (the difference between the smallest relaxed solutions and the best feasible solution) was equal to 1, and the corresponding relative gaps (the relative differences between the smallest relaxed solutions and the best feasible solutions) were less than 2.4 percent.

Table 3.3: Number of secondary structure instances solved by A_purva

	SSEs
similar instances (164)	**164**
dissimilar instances (616)	613
Total (780)	777

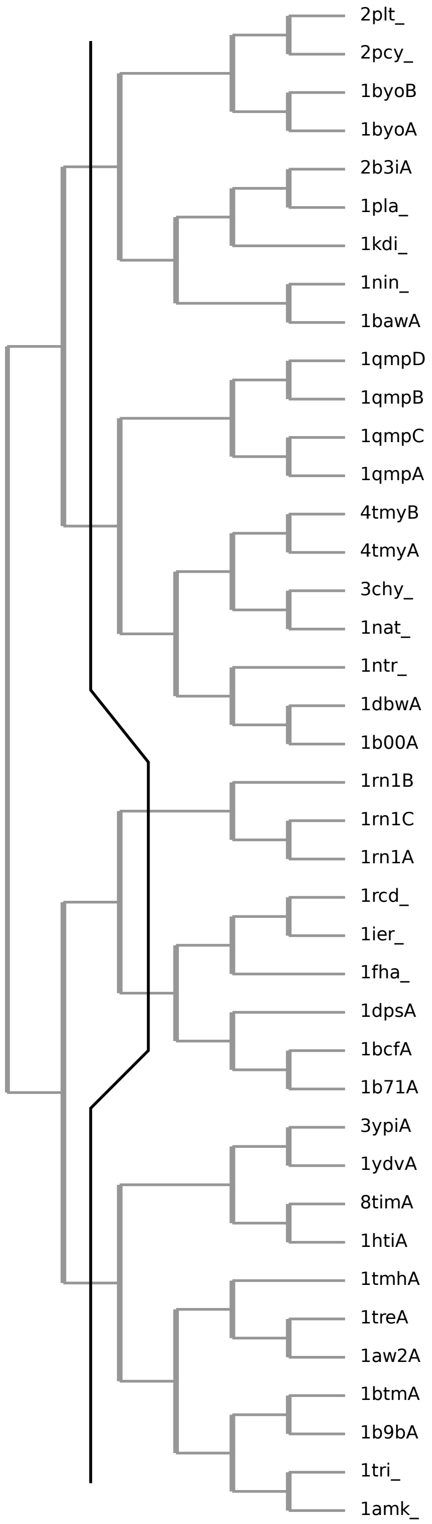
Time limit was fixed to 1 sec. per secondary structure instance. A_purva succeeded in solving 777 instances. Unsolved instances are : 1ninA–3ypiA, 1ninA–1treA, 1ninA–8timA.

Second, we wanted to know if the secondary structure alignments found during the previous experiment can be used for automatic classification. In order to take into account the weights on the edges, the similarity function from [69] was modified as follows:

$$SIM(P_1, P_2) = \frac{2 \times LB}{\sum_{(i,j)\in E_1} w1_{ij} + \sum_{(k,l)\in E_2} w2_{kl}}. \tag{3.11}$$

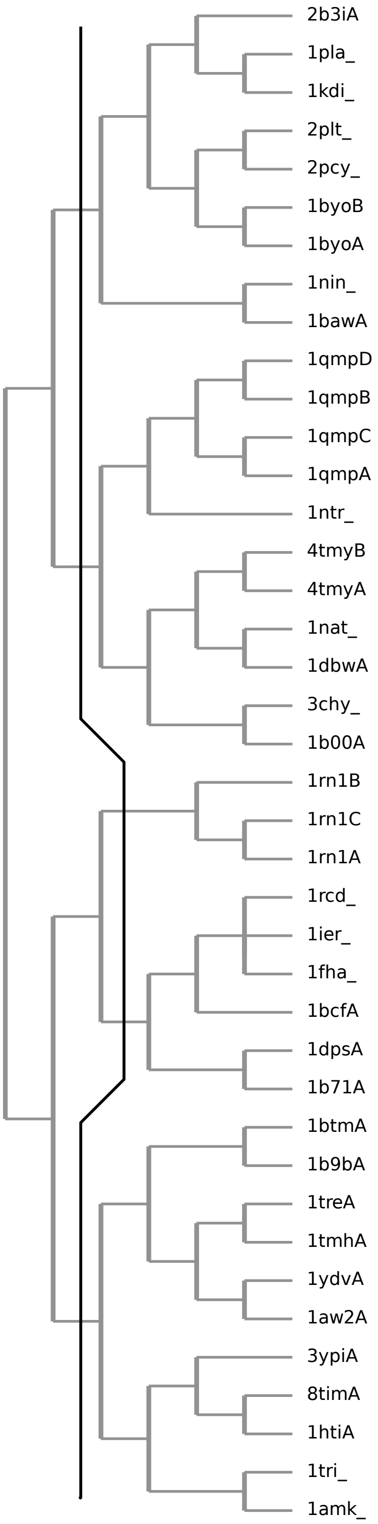
This similarity index was given to Chavl, which returned both an ascendant classification and the cut corresponding to the best partition (both are presented in **figure 3.9**). The total running time was about 4.10 sec (including the three instances which were not optimally solved, and which each required 1 second). It is about 49 times

Figure 3.7: Ascendant classification obtained when using SSE filter 1.



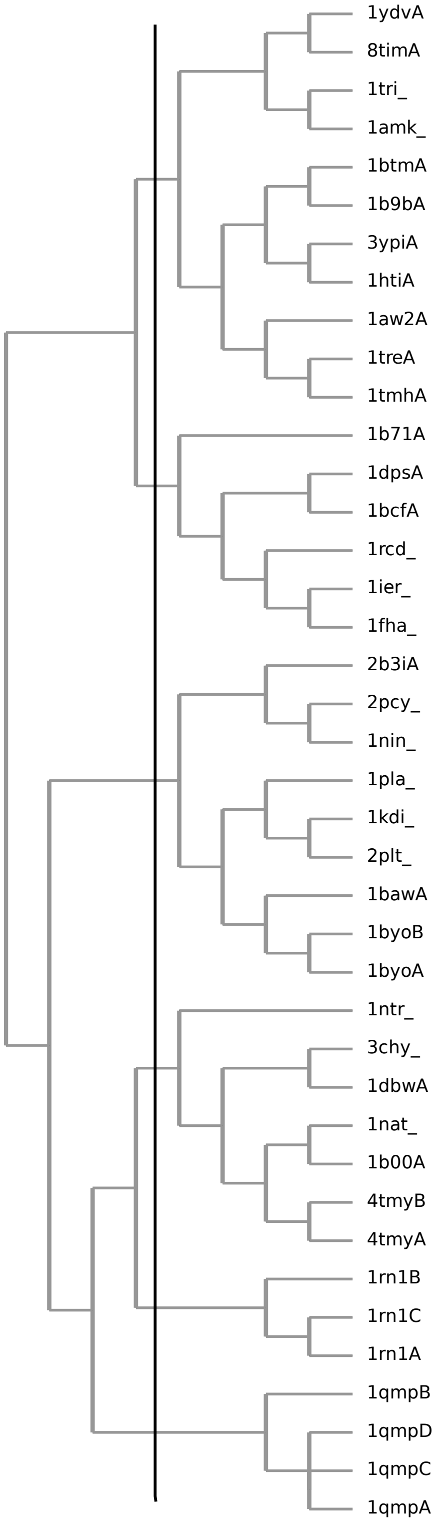
Ascendant classification returned by Chavl when using the scores computed by A_purva_SSE1. Chavl also returns the cut corresponding to the best partition.

Figure 3.8: Ascendant classification obtained when using SSE filter 2.



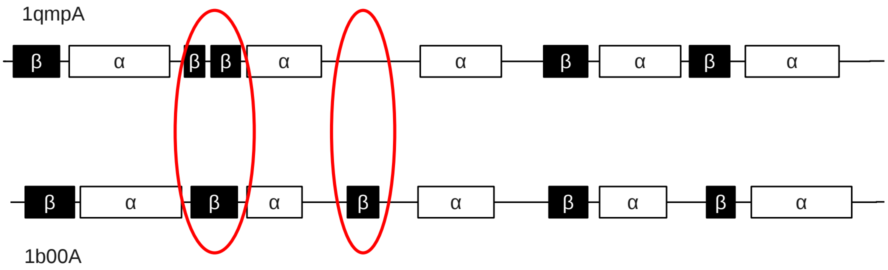
Ascendant classification returned by Chavl when using the scores computed by A_purva_SSE2. Chavl also returns the cut corresponding to the best partition.

Figure 3.9: Ascendant classification obtained by using SSE alignment only.



Ascendant classification returned by Chavl when using scores based on secondary structure contact map alignments. Chavl also returns the cut corresponding to the best partition.

Figure 3.10: SSE assignments of 1qmpA and 1b00A.



Protein chains 1qmpA and 1b00A are from the same SCOP family (CheY-related). However, SSE assignments returned by Kaksi show two differences: the second β-strand, present in all members of the family, is split in two in 1qmpA, and second, the third β-strand, which is in the middle of the β-sheet, is not detected in 1qmpA. These two "errors" are also present in the SSE assignments of 1qmpB, 1qmpC and 1qmpD.

Table 3.4: Automatic classification obtained by using secondary structure alignments.

Classes	Proteins	SCOP Family
1	1qmp(A,B,C,D)	CheY-related
2	1b00A, 1dbwA, 1natA, 1ntrA, 3chyA, 4tmy(A,B)	CheY-related
3	1bawA, 1byo(A,B), 1kdiA, 1ninA 1plaA, 2b3iA, 2pcyA, 2pltA	Plastocyanin /azurin-like
4	1amkA, 1aw2A, 1b9bA, 1btmA, 1htiA 1tmhA, 1treA, 1triA, 1ydvA, 3ypiA, 8timA	Triosephosphate isomerase (TIM)
5	1b71A, 1bcfA, 1dpsA, 1fhaA, 1ierA, 1rcdA	Ferritin
6	1rn1(A,B,C)	Fungal ribonucleases

Automatic classification returned by Chavl when using the similarity score associated to secondary structure alignments. The only difference with the SCOP one at family level is that the CheY-related family was split into two classes.

faster than the 201.21 sec. needed when using SSE filter 2, and about 72.6 times faster than the 297.67 sec. needed whithout filtering.

The corresponding classification is presented in **table 3.4**, and it is very close to the SCOP one at family level, the only difference being that the SCOP family "CheY-related" is divided in two. This "error" is introduced by Kaksi which does not assign correctly two β-strands in protein chains 1qmpA, 1qmpB, 1qmpC and 1qmpD, as shown in **figure 3.10**. These two SSEs are parts of a β-sheet (in this case a set of five parallel β-strands) and are involved in many contacts.

4.3 Hierarchical CMO

In this section, we compare A_purva to its hierarchical counterpart both in terms of number of solved instances and in terms of running times.

Solved instances comparison

We used A_purva and A_purva_H to solve the 780 Skolnick set instances, when setting a time limit of 1800 seconds per instance. Note that for A_purva_H, this time limit includes both the time needed for solving the secondary structure alignments and the time needed for solving the amino-acid alignments. As illustrated in **table 3.5**, the hierarchical version of A_purva succeeded to solve 134 instances more than the original A_purva.

Table 3.5: Number of solved instances comparison

	A_purva	SSEs	A_purva_H
Similar instances (164)	**164**	164	163
Dissimilar instances (616)	465	614	**600**
Total (780)	629	778	**763**

Number of solved instances from the Skolnick set, when the time limit is fixed to 1800 sec. per instance. The easy instances are instances such that both protein structures come from the same SCOP family, and dissimilar instances are instances such that the two protein structures come from different SCOP families. A_purva_H (4^th column) is able to solve 134 more instance than A_purva (2^nd column). The number of Secondary structure alignment instances solved during the first step of A_purva_H is also presented (3^rd column).

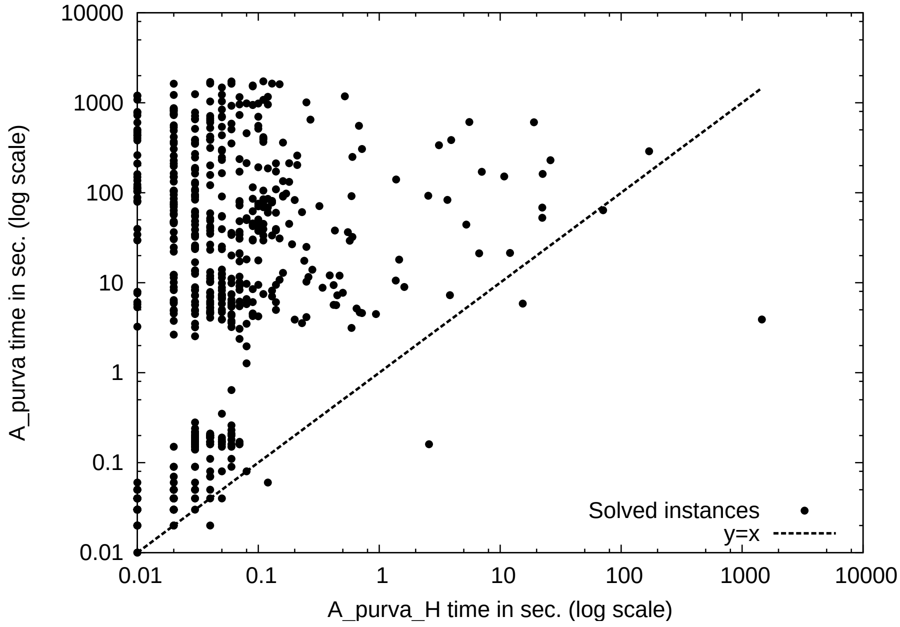
Running times comparison

Figure 3.11 compares the running time of A_purva to the one of the A_purva_H on the set of 613 Skolnick instances solved by both algorithms. A_purva needed 99976.55 sec. to solve these instances while A_purva_H only needed 1941.10 sec., and thus A_purva_H is about 51.51 times faster than A_purva, and this difference goes up to 120329 times (for instance 1bcfA-1rn1C, solved by A_purva in 1203.29 sec. and by A_purva_H in 0.01 sec.).

5 Conclusion

In order to accelerate A_purva, we proposed to reduce the size of the amino-acid alignment graphs by using two filters based on the secondary structure of the proteins. Using these filters allows A_purva to be about 50 times faster. The obtained results motivated us to further exploit the secondary structure informations and to propose a hierarchical approach for solving CMO, which consists in first aligning secondary structure elements, and then to use this secondary structure alignment for filtering the amino-acid alignment. Towards this goal we developed a new secondary structure alignment approach based on weighted contact maps. This weighted CMO problem is efficiently solved by a modified version of A_purva, and we show that the secondary structure alignments it returns can be used for obtaining a very fast automatic classification of proteins, except for proteins for which secondary structures are not assigned correctly.

Figure 3.11: time comparison



For each of the 613 Skolnick instance solve by both methods, the running time of A_purva_H (x-axis) is plotted against the running time of A_purva (y-axis). In average, the hierarchical version of A_purva is about 51.51 times faster than the original A_purva (up to 120329 times).

Chapter 4

Improvement of the contact map approach: a maximum clique problem

1 Problematics

One of the main weakness of CMO is that in order to maximize the number of common contacts, it also introduces some "errors" like aligning two amino-acids that are close in 3D space with two amino-acids that are remote, as illustrated in **figure 4.1**. These errors could potentially yield alignments with large root mean square deviations (RMSD) which is not desirable for structure comparisons.

The results of Godzik, taken from [27] and presented in table 4.1, illustrate this RMSD problem. **Table 4.1** compares the alignments returned by CMO to the ones returned by DALI and by GRAFIT. While the alignments returned by CMO are longer (in terms of amino-acids), they also possess the largest $RMSD_c$ values.

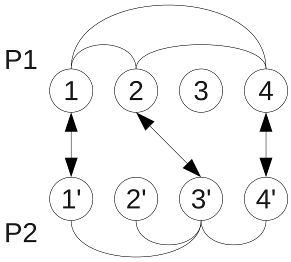
The goal of the work presented in this chapter is to avoid such problems by reformulating the alignment between two protein structures.

2 Distance-based Alignment Search Tool (DAST)

As previously illustrated, CMO only tries to maximize the number of common contacts, without taking into account the remote amino-acids. Matching close amino-acids only with close amino-acids, and remote amino-acids only with remote amino-acids does not really solve this problem, since two pairs of remote amino-acids are not necessarily comparable.

This motivated us to replace the notion of common contact by the more general notion of similar internal distance (according to a fixed threshold), and then to propose DAST (Distance-based Alignment Search Tool), a protein structure alignment method based on internal distances which is modeled as a maximum clique problem in alignment graphs. In DAST, the two proteins P_1 and P_2 are represented by their ordered sets of amino-acids N_1 and N_2. In order to reduce the size of the alignment graph, we use the rule presented in the previous chapter for the SSE filter 2: two amino-acids $i \in N_1$ and $k \in N_2$ are compatible only if they come from the same kind of secondary structure elements (i.e. i and k both come from an α-helix, or

Figure 4.1: An optimal CMO matching.



Two proteins (P_1 and P_2) are represented by their contact map graphs where the vertices correspond to the amino-acids and where edges connect amino-acids in contacts (i.e. close). The matching "$1 \leftrightarrow 1', 2 \leftrightarrow 3', 4 \leftrightarrow 4'$", represented by the arrows, yields two common contacts which is the maximum for the considered case. However, it also matches amino-acids 1 and 4 from P_1 which are in contacts with amino-acids $1'$ and $4'$ in P_2 which are remote.

Table 4.1: Comparison of published alignments for two pairs of protein structures.

Instance	Method	$RMSD_c$ (Å)	nb contacts	nb identities	length
1azcA–1plcA	CMO	5.1	**180**	**15**	89
	DALI	4.5	68	13	**93**
	GAFIT	**1.5**	65	11	60
4fxnA–3chyA	CMO	4.3	**107**	**14**	108
	DALI	3.8	44	11	**112**
	GAFIT	**2.9**	72	12	100

For the two protein structure comparison instances, the $RMSD$ between the aligned amino-acid's α-carbons ($RMSD_c$) is presented in the third column, the number of common contact is presented in the fourth column, the number of amino-acid identities is presented in the fifth column and the length of the alignment (in amino-acid) is presented in the last column. CMO alignments are longer (in terms of amino-acids), but they also possess the biggest $RMSD_c$ values.

from a β-strand) or if both come from a loop. Let us denote by d_{ij} (resp. $d_{k.l}$) the euclidean distance between the α-carbons of amino-acids i and j (resp. k and l). Matching $i \leftrightarrow k$ is compatible with matching $j \leftrightarrow l$ only if $|d_{ij} - d_{kl}| \leq \tau$, where τ is a distance threshold. The longest alignment in terms of amino-acids, in which each couple of amino-acids from P_1 is matched with a couple of amino-acids from P_2 having similar distance relations, corresponds to a maximum clique in G.

Since $RMSD_d = \sqrt{\frac{1}{N_m} \times \sum(|d_{ij} - d_{kl}|^2)}$, where N_m is the number of matching pairs "$i \leftrightarrow k, j \leftrightarrow l$", the alignments given by DAST have a $RMSD_d$ of internal distances $\leq \tau$.

3 Integer programming model

By using the properties of our alignment graphs, we designed a new integer programming model (whose formulation is very different from [5, 53]) for solving the maximum weighted clique problem (and thus the maximum clique problem), where the weights W_{ik} associated to the vertices $i.k$ and W_{ikjl} associated to the edges $(i.k, j.l)$ are all in $\mathbb{R}$,

To each vertex $i.k \in V$ (in row $i \in V_1$ and column $k \in V_2$), we associate a binary variable x_{ik} such that :

$$x_{ik} = \begin{cases} 1 & \text{if vertex } i.k \text{ is in the clique,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

In the same way, to each edge $(i.k, j.l) \in E$, we associate a binary variable y_{ikjl} such that :

$$y_{ikjl} = \begin{cases} 1 & \text{if edge } (i.k, j.l) \text{ is in the clique,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

The goal is to find a clique which maximizes the sum of its vertex weights and the sum of its edge weights. This leads to the objective function :

$$Z_{MWC} = \max \sum_{i.k} W_{ik}\ x_{ik} + \sum_{(i.k,j.l)} W_{ikjl}\ y_{ikjl}. \tag{4.3}$$

The one-to-one matching implies special order set constraints. In each row $i \in V_1$, at most one vertex can be chosen (only one x_{ik} can be set to 1).

$$\sum_{k} x_{ik} \leq 1, \quad \forall i \in V_1. \tag{4.4}$$

The same holds for the columns.

$$\sum_{i} x_{ik} \leq 1, \quad \forall k \in V_2. \tag{4.5}$$

These special order set constraints (at most one activated vertex per row and per column) lead to compact formulations of the relations between vertices and edges. Denote by $d^+_{col}(i.k)$ the set of columns l, $l > k$, such that $\exists (i.k, j.l) \in E$. In a similar way, $d^-_{col}(i.k)$ is the set of columns l, $l < k$, such that $\exists (j.l, i.k) \in E$. $d^+_{row}(i.k)$ is the set of rows j, $j > i$, such that $\exists(i.k, j.l) \in E$. And finally, $d^-_{row}(i.k)$ is the set of rows j, $j < i$, such that $\exists(j.l, i.k) \in E$.

Edges driven activations of vertices can be formulated with the following compact inequalities :

$$x_{ik} \geq \sum_j y_{ikjl}, \quad \forall i.k \in V,\ \forall l \in d^+_{col}(i.k). \tag{4.6}$$

$$x_{jl} \geq \sum_i y_{ikjl}, \quad \forall j.l \in V,\ \forall k \in d^-_{col}(j.l). \tag{4.7}$$

$$x_{ik} \geq \sum_l y_{ikjl}, \quad \forall i.k \in V,\ \forall j \in d^+_{row}(i.k). \tag{4.8}$$

$$x_{jl} \geq \sum_k y_{ikjl}, \quad \forall j.l \in V,\ \forall i \in d^-_{row}(j.l). \tag{4.9}$$

Vertices driven activations of edges can be formulated with the following compact inequalities :

$$\sum_i x_{ik} + \sum_j x_{jl} - \sum_{ij} y_{ikjl} \leq 1,\ \forall k \in V_2,\ \forall l \in V_2,\ k < l. \tag{4.10}$$

$$\sum_k x_{ik} + \sum_l x_{jl} - \sum_{kl} y_{ikjl} \leq 1,\ \forall i \in V_1,\ \forall j \in V_1,\ i < j. \tag{4.11}$$

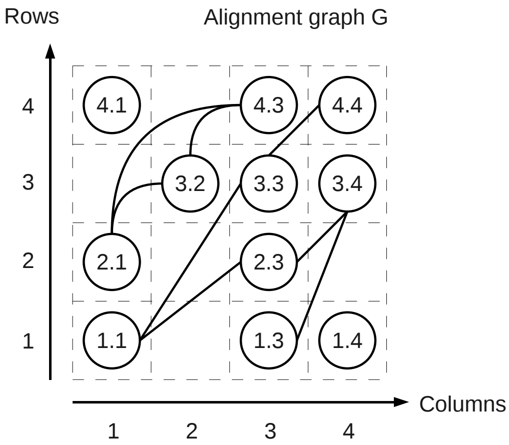
4 Branch and Bound approach

We have been inspired by [52] to propose our own algorithm which is more suitable for solving the maximum clique problem in the previously defined $m \times n$ alignment graph $G = (V, E)$. Note that the next illustrations are based on the alignment graph presented in **figure 4.2**.

As written in the first chapter, a successor of a vertex $i.k \in V$ is an element of the set $\Gamma^+(i.k) = \{j.l \in V \text{ s.t. } (i.k, j.l) \in E, i < j \text{ and } k < l\}$. $V^{i.k}$ is the subset of V restricted to vertices in rows j, $i \leq j \leq m$, and in columns l, $k \leq l \leq n$. $\Gamma^+(i.k) \subset V^{i+1.k+1}$. $G^{i.k}$ is the subgraph of G induced by the vertices in $V^{i.k}$, and the cardinality of a vertex set U is $|U|$.

Let *Best* be the biggest clique found so far (first it is set to $\emptyset$), and $|\overline{MCC}(G)|$ be an over-estimation of the cardinality of a maximum clique ($|MCC(G)|$). By definition, $V^{i+1.k+1} \subset V^{i.k+1} \subset V^{i.k}$, and similarly $V^{i+1.k+1} \subset V^{i+1.k} \subset V^{i.k}$. From these inclusions and from the definition of an alignment graph, it is easily seen that for any $G^{i.k}$, $MCC(G^{i.k})$ is the biggest clique among $MCC(G^{i+1.k})$, $MCC(G^{i.k+1})$ and $MCC(G^{i+1.k+1}) \bigcup \{i.k\}$, but for the latter only if vertex $i.k$ is adjacent to all vertices in $MCC(G^{i+1.k+1})$. Let C be a $(m + 1) \times (n + 1)$ array where $C[i][k] = |\overline{MCC}(G^{i.k})|$ (values in row $m + 1$ or column $n + 1$ are equal to 0). For reasoning purpose, let assume that the upper-bounds in C are exact. If a vertex $i.k$ is adjacent to all vertices in $MCC(G^{i+1.k+1})$, then $C[i][k] = 1 + C[i + 1][k + 1]$, else $C[i][k] = \max(C[i][k + 1], C[i + 1][k])$. We can deduce that a vertex $i.k$ cannot be in a clique in $G^{i.k}$ which is bigger than *Best* if $C[i + 1][k + 1] < |Best|$, and this reasoning still holds if values in C are upper estimations. Another important inclusion is

Figure 4.2: A 4 × 4 alignment graph.

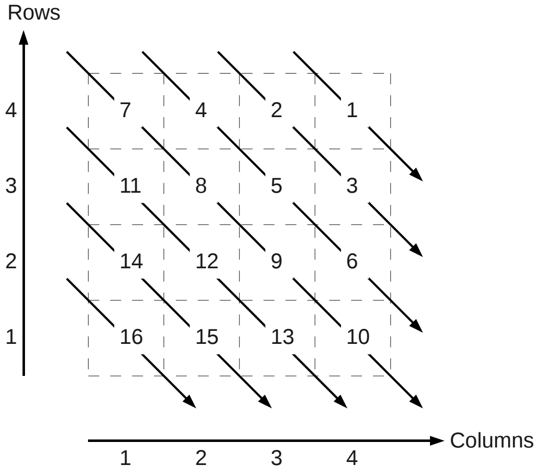


$\Gamma^+(i.k) \subset V^{i+1.k+1}$. Even if $C[i+1][k+1] \geq |Best|$, if $|\overline{MCC}(\Gamma^+(i.k))| < |Best|$ then $i.k$ cannot be in a clique in $G^{i.k}$ bigger than $Best$.

Our main clique cardinality estimator is constructed and used according to these properties. A function, Find_clique(G), will visit the cells of T according to north-west to south-est diagonals, from diagonal "$i + k = m + n$" to diagonal "$i + k = 2$" as illustrated in **figure 4.3**. For each cell $T[i][k]$ containing a vertex $i.k \in V$, it may call Extend_clique($\{i.k\}$, $\Gamma^+(i.k)$), a function which tries to extend the clique $\{i.k\}$ with vertices in $\Gamma^+(i.k)$ in order to obtain a clique bigger than $Best$ (which cannot be bigger than |Best| +1). If such a clique is found, $Best$ is updated. However, Find_clique() will call Extend_clique() only if two conditions are satisfied : (i) $C[i+1][k+1] = |Best|$ and (ii) $|\overline{MCC}(\Gamma^+(i.k))| \geq |Best|$. After the call to Extend_clique(), $C[i][k]$ is set to $|Best|$. For all other cells $T[i][k]$, $C[i][k]$ is set to $\max(C[i][k+1], C[i+1][k])$ if $i.k \notin V$, or to $1 + C[i+1][k+1])$ if $i.k \in V$. Note that the order used for visiting the cells in T guaranties that when computing the value of $C[i][k]$, the values of $C[i+1][k]$, $C[i][k+1]$ and $C[i+1][k+1]$ are already computed.

Array C can also be used in function Extend_clique() to fasten the maximum clique search. This function is a branch a bound (B&B) search using the following branching rules. Each node of the B&B tree is characterized by a couple (Cli, $Cand$) where Cli is the clique under construction and $Cand$ is the set of candidate vertices to be added to Cli. Each call to Extend_clique($\{i.k\}$, $\Gamma^+(i.k)$) create a new B&B tree which root node is ($\{i.k\}$, $\Gamma^+(i.k)$). The successors of a B&B node $(Cli, Cand)$ are the nodes $(Cli \bigcup \{i'.k'\}$, $Cand \bigcap \Gamma^+(i'.k'))$, for all vertices $i'.k' \in Cand$. Branching follows lexicographic increasing order (row first). According to the branching rules, for any given B&B node (Cli, $Cand$) the following cutting rules holds : (i) if $|Cli| + |Cand| \leq |Best|$ then the current branch cannot lead to a clique bigger than $|Best|$ and can be fathomed, (ii) if $|\overline{MCC}(Cand)| \leq |Best| - |Cli|$, then the current branch cannot lead to a clique bigger than $|Best|$, and (iii) if

Figure 4.3: Visiting order of array T.



$|\overline{MCC}(Cand \bigcap \Gamma^+(i.k))| \leq |Best| - |Cli| - 1$, then branching on $i.k$ cannot lead to a clique bigger than $|Best|$. For any set $Cand$ and any vertex $i.k$, $Cand \bigcap \Gamma^+(i.k) \subset \Gamma^+(i.k)$, and $\Gamma^+(i.k) \subset G^{i+1.k+1}$. From these inclusions we can deduce two way of over-estimating $|MCC(Cand \bigcap \Gamma^+(i.k))|$. First, by using $C[i+1][k+1]$ which over-estimates $|MCC(G^{i+1.k+1})|$ and second, by over-estimating $|MCC(\Gamma^+(i.k))|$. All values $|\overline{MCC}(\Gamma^+(i.k))|$ are computed once for all in Find_clique() and thus, only $|\overline{MCC}(Cand)|$ needs to be computed in each B&B node.

4.1 Maximum clique cardinality estimators

Even if the described functions depend on array C, they also use another upper-estimator of the cardinality of a maximum clique in an alignment graph. By using the properties of alignment graphs, we developed the following estimators.

Row and columns numbers

Definition 6 implies that there is no edge between vertices from the same row or the same column. This means that in a $m \times n$ alignment graph, $|MCC(G)| \leq \min(m, n)$. If the numbers of rows and columns are not computed at the creation of the alignment graph, they can be computed in $O(|V|)$ times.

Longest increasing subset of vertex

Definition 11 *An* ***increasing subset of vertices*** *in an alignment graph $G = \{V, E\}$ is an ordered subset $\{i_1.k_1,\ i_2.k_2,\ \ldots,\ i_t.k_t\ \}$ of V, such that $\forall j \in [1, t-1]$, $i_j < i_{j+1}$, $k_j < k_{j+1}$. $LIS(G)$ is the longest, in terms of vertices, increasing subset of vertices of G.*

Since any two vertices in a clique are adjacent, the definition of an alignment graph implies that a clique in G is an increasing subset of vertices. However, an increasing subset of vertices is not necessarily a clique (since vertices are not necessarily adjacent), and thus $|MCC(G)| \leq |LIS(G)|$. In a $m \times n$ alignment graph $G = (V, E)$, $LIS(G)$ can be computed in $O(n \times m)$ times by dynamic programming. However, it is possible by using the longest increasing subsequence to solve $LIS(G)$ in $O(|V| \times \ln(|V|))$ times which is more suited in the case of sparse graph like in our protein structure comparison experiments.

Definition 12 *The* ***longest increasing subsequence*** *of an arbitrary finite sequence of integers $S =$ "$i_i, i_2, \ldots, i_n$" is the longest subsequence $S' =$ "$i'_i, i'_2, \ldots, i'_t$" of S respecting the original order of S, and such that for all $j \in [1, t], i'_j < i'_{j+1}$. By example, the longest increasing subsequence of "1,5,2,3" is "1,2,3".*

For any given alignment graph $G = \{V, E\}$, we can easily reorder the vertex set V, first by increasing order of columns, and second by decreasing order of rows. Let's denote by V' this reordered vertex set. Then we can create an integer sequence S corresponding to the row indexes of vertices in V'. For example, by using the alignment graph presented in Figure 4.2, the reordered vertex set V' is {4.1, 2.1, 1.1, 3.2, 4.3, 3.3, 2.3, 1.3, 4.4, 3.4, 1.4}, and the corresponding sequence of row indexes S is "4, 2, 1, 3, 4, 3, 2, 1, 4, 3, 1". An increasing subsequence of S will pick at most one number from a column, and thus an increasing subsequence is longest if and only if it covers a maximal number of increasing rows. This proves that solving the longest increasing subsequence in S is equivalent to solving the longest increasing subset of vertices in G. Note that the longest increasing subsequence problem is solvable in time $O(l \times \ln(l))$ [19], where l denotes the length of the input sequence. In our case, this corresponds to $O(|V| \times \ln(|V|))$.

Longest increasing path

Definition 13 *An* ***increasing path*** *in an alignment $G = \{V, E\}$ is an increasing subset of vertex $\{i_1.k_1, i_2.k_2, \ldots, i_t.k_t\}$ such that $\forall j \in [1, t-1]$, $(i_j.k_j, i_{j+1}.k_{j+1}) \in E$. The longest increasing path in G is denoted by $LIP(G)$*

As the increasing path take into account edges between consecutive vertices, $|LIP(G)|$, should better estimate $|MCC(G)|$. $|LIP(G)|$ can be computed in $O(|V|^2)$ by the following recurrence. Let $DP[i][k]$ be the length of the longest increasing path in $G^{i.k}$ containing vertex $i.k$. $DP[i][k] = 1 + \max_{i'.k' \in \Gamma^+ i.k}(DP[i'][k'])$. The sum over all $\Gamma^+(i.k))$ is done in $O(|E|)$ time complexity, and finding the maximum over all $DP[i][k]$ is done in $O(|V|)$. This results in a $O(|V| + |E|)$ time complexity for computing $|LIP(G)|$.

Amongst all of the previously defined upper bounds, the longest increasing subset of vertices (solved using the longest increasing subsequence) exhibits the best performances and is the one we used for obtaining the results presented in the next section.

5 Comparison with related protein structure comparison methods

Three methods from the literature are highly related to DAST, in the sense that they try to find a protein structure alignment which minimize the internal-distance differences induced by the matched alpha-carbon. The first one, Dali [35, 37], was proposed by Holm and Sander in 1993. The second one, FAST [71], was proposed by Zhu and Wang in 2005. The last one, Paul [66, 67], was recently developped by Wohlers, Petzold, Domingues and Klau in 2009.

Dali: No matter if Dali uses rigid or elastic scores, it does not prohibit matching pairs that introduce big distance differences (higher than the specified threshold τ). Instead, they are rewarded negatively in the objective function, and thus a matching pair $i \leftrightarrow k$ can be kept in the alignment if its negative contributions are counterbalanced by positive ones. For this reason, Dali does not guaranty that the RMSDs of its alignments are less than a fixed threshold. From an algorithmic point of view, Dali do not use an exact solver for its maximum edge-weighted clique problem, but uses a heuristic (Monte-Carlo).

Fast: It can be seen as Dali when using order-preserving, even if its edge scoring scheme is not exactly equal to Dali's elastic one, as it also takes into account orientations of amino-acid R groups (also called sidechains). Similarly to Dali, FAST does not guaranty that the RMSDs of the returned alignments are less than a fixed threshold. However, FAST introduces an interesting heuristic filter for removing "bad" matching pairs $i \leftrightarrow k$ based on five consecutive amino-acids fragments. Matching pair $i \leftrightarrow k$ is prohibited (i.e. vertex $i.k \notin V$) if the three internal distances $d_{i-2,i+1}$, $d_{i-2,i+2}$ and $d_{i-1,i+2}$ in P_1 are not similar to their counterparts $d_{k-2,k+1}$, $d_{k-2,k+2}$ and $d_{k-1,k+2}$ in P_2.

Paul: Based on the same observation which led to the creation of DAST (that maximising the number of common contacts may introduce large internal-distance differences), Paul is a CMO based method in which the notion of common contact is replaced by the one of similar internal distances. If the two proteins P_1 and P_2 are represented by their ordered sets of amino-acids N_1 and N_2, Paul can be modelled as an $|N_1| \times |N_2|$ alignment graph in the following way. All amino-acids from P_1 are compatible with all amino-acids from P_2, and thus, for all $i \in N_1$ and all $k \in N_2$, vertex $i.k$ is in V. Paul tries to find an order preserving matching and consequently, the matching pair $i \leftrightarrow k$ is compatible with the matching pair $j \leftrightarrow l$ only if $i < j$ and $k < l$. Edge $(i.k, j.l)$ exist is matching pairs $i \leftrightarrow k$ and $j \leftrightarrow l$ are compatible and if $|d_{ij} - d_{kl}|$ is less than a given threshold and if the number of amino-acids between i and j is similar to the one between k and l (according to the sequences of P_1 and p_2). To each edge $(i.k, j.l) \in E$ is associated a weight based on the rigid score of Dali. To each vertex $i.k \in V$ is associated a negative weight which is used to avoid introducing too many large internal distance differences (since the negative vertex weights are not compensated by enough positive edge weights). The optimal alignment is then to find in G the maximum weighted feasible path (including both vertex and edge weights) . Since Paul does not look in G for a clique, the alignments it returns may contain matching pairs which introduce large internal distance differances, and thus Paul does not guaranty that the RMSDs of its alignments are less than a fixed threshold.

Table 4.2: Characteristics of the alignment graphs.

	Number of vertices	Number of edges	Density
Set name	min, average, max	min, average, max	min, average, max
Skolnick	100, 158.92, 208	886, 2368.69, 3547	0.16, 0.18, 0.20

6 Computational results

All results presented in this section come from real protein structure comparison instances. The integer programming model has been implemented using ILOG Cplex 10 library and is denoted by *MIP*. Our branch and bound algorithm has been implemented in C and is denoted by *ACF* (for Alignment Clique Finder).

Both were compared to the original VAST clique solver which is based on Bron and Kerbosch's algorithm [9] (denoted by *BK*). It is important to note that *BK* is not a maximum clique finder but it returns all maximal cliques in a graph. *BK* is now outperformed by more recent algorithms based on smart pivot selection [15]. *BK* is presented here because it was already integrated in VAST. *ACF* was also compared to Östergård's algorithm [52] (denoted by *Östergård*), which is one of the fastest maximum clique finders from the literature. All algorithms were set to solve the maximum clique problem.

6.1 MIP solver on SSE alignment instances

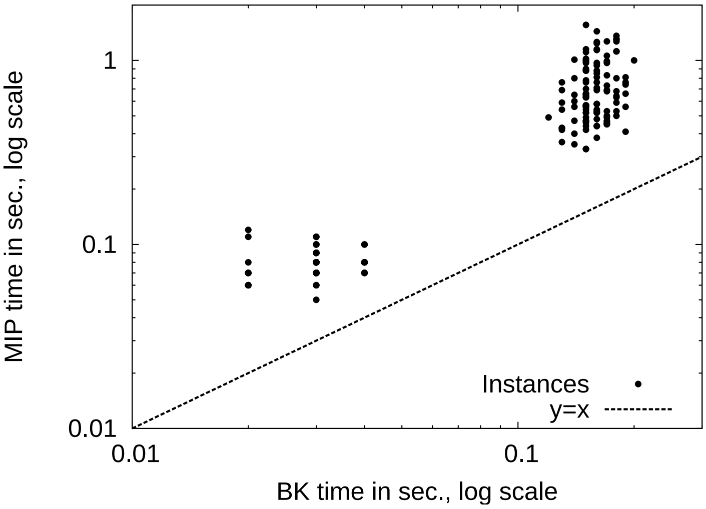
In this section, MIP is compared to BK in the context of secondary structure alignment in VAST. The protein alignment instances come from the Skolnick set, but we only use the 170 instances for which the alignment graphs contain more than 100 vertices. The corresponding graphs are described in table 4.2.

Figure 4.4 compares the time needed by MIP to the one of BK on the 170 Skolnick instances. In average, MIP is 3.35 times slower than BK. This is not surprising, since dedicated solvers are expected to be faster than general purpose solvers (CPLEX in this case). This observation motivated us to go further in developing a fast special purpose clique solver.

6.2 ACF on SSE alignment instances

This section illustrates the behavior of *ACF* in the context of secondary structure element (SSE) alignments. For this purpose we integrated *ACF* and *Östergård* (whose code is freely available) in VAST. We afterwards compared them with *BK* by selecting few large protein chains having between 80 to 90 SSE's (for small protein chains the running times of both *Östergård* and *ACF* are less than 0.01 sec.). Computations were done on a AMD at 2.4 GHz computer, and the corresponding running times are presented in **table 4.3**. We observe that *Östergård* is 4053 times faster than *BK*, and that *ACF* is about 9.3 times faster than *Östergård*. Although we have chosen large protein chains, the SSE alignment graphs are relatively small (up to 5423 vertices and 551792 edges). On such graphs the difference between *Östergård* and *ACF* performance is not very visible–it will be better illustrated on

Figure 4.4: MIP vs BK running time comparison on Skolnick set.



For each instance the execution time of MIP is plotted on the x-axis, while the one of BK is depicted on the y-axis. All points are above the $x = y$ line (i.e. BK is always faster than MIP).

larger alignment graphs in the next section.

6.3 ACF on DAST alignment instances

In this section we compare *ACF* to *Östergård* in the context of amino-acid alignments in DAST. Computations were done on a PC with an Intel Core2 processor at 3Ghz, and for both algorithms the computation time was bound to 5 hours per instance. Secondary structures assignments were done with KAKSI [45], and the threshold distance τ was set to 3Å. The protein structures come from the Skolnick set, described in [41]. It contains 40 protein chains having from 90 to 256 amino-acids, classified in SCOP [4] (v1.73) into five families. Amongst the 780 corresponding alignment instances, 164 comes from the same family and will be called "similar". The 616 other instances come from different families and thus will be called

Table 4.3: Running time comparison on secondary structure alignment instances.

Instances		BK (sec.)	Östergård (sec.)	ACF (sec.)
1k32B	1n6eI	1591.89	1.42	**0.09**
1k32B	1n6fB	1546.78	**0.01**	**0.01**
1k32B	1n6fF	1584.25	0.14	**0.02**
1n6dD	1k32B	1373.35	0.06	**0.01**
1n6dD	1n6eI	1390.27	0.11	**0.03**
1n6dD	1n6fB	1328.85	0.65	**0.06**
1n6dD	1n6fF	1398.41	0.13	**0.05**

Running time comparison of *BK*, *Östergård* and *ACF* on secondary structure alignment instances for long protein chains (containing from 80 to 90 SSE's). *BK* is notably slower than the *Östergård*'s algorithm, which is slightly slower than *ACF*.

Table 4.4: DAST alignment graphs characteristics.

		array size	|V|	|E|	density	|MCC|
similar instances	min	97×97	4018	106373	8.32%	45
	max	256×255	25706	31726150	15.44%	233
dissimilar instances	min	97×104	1581	77164	5.76%	12
	max	256×191	21244	16839653	14.13%	48

All alignment graphs from DAST have small edge density (less than 16%). Similar instances are characterized by bigger maximum cliques than the dissimilar instances.

Table 4.5: Number of solved instances comparison.

	Östergård	ACF
Similar instances (164)	128	**155**
Dissimilar instances (616)	545	**616**
Total (780)	673	**771**

Number of solved instances on Skolnick set: *ACF* solves 21% more similar instances and 13% more dissimilar instances than *Östergård*.

"dissimilar". Characteristics of the corresponding alignment graphs are presented in **table 4.4**.

Table 4.5 compares the number of instances solved by each algorithm on Skolnick set. *ACF* solved 155 from 164 similar instances, while *Östergård* solved 128 instances. *ACF* was able to solve all 616 dissimilar instances, while *Östergård* solved 545 instances only. Thus, on this popular benchmark set, *ACF* clearly outperformed *Östergård* in terms of number of solved instances.

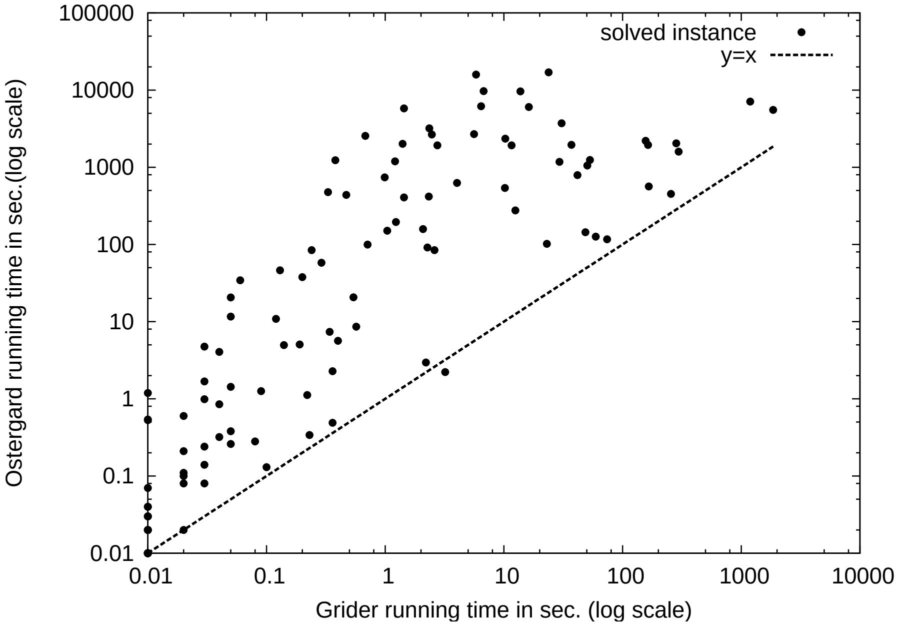
Figure 4.5 compares the running time of *ACF* to the one of *Östergård* on the set of 164 similar instances solved by both algorithms (all instances solved by *Östergård* were also solved by *ACF*). For all instances except one, *ACF* is significantly faster than *Östergård*. **Figure 4.6** compares the running time of *ACF* to the one of *Östergård* on the set of 509 dissimilar instances solved by both algorithms (again, all instances solved by *Östergård* were also solved by *ACF*). For all instances, *ACF* is significantly faster than *Östergård*. More precisely, *ACF* needed 12 hs. 29 min. 56 sec. to solve all these 673 instances, while *Östergård* needed 260 hs. 10 min. 10 sec. Thus, on the Skolnick set, *ACF* is about 20 times faster in average than *Östergård* (up to 4029 times for some instances).

7 Comparison between CMO and DAST

The purpose of this section is to confirm experimentally that DAST provides alignments with good (i.e. small) $RMSD_d$ values. Towards this goal, we extracted ten instances from the Skolnick set. The SSEs assignments were obtained by Kaksi. The internal distance threshold of DAST was set to 3 Å, and the contact threshold of CMO was set to 7.5 Å.

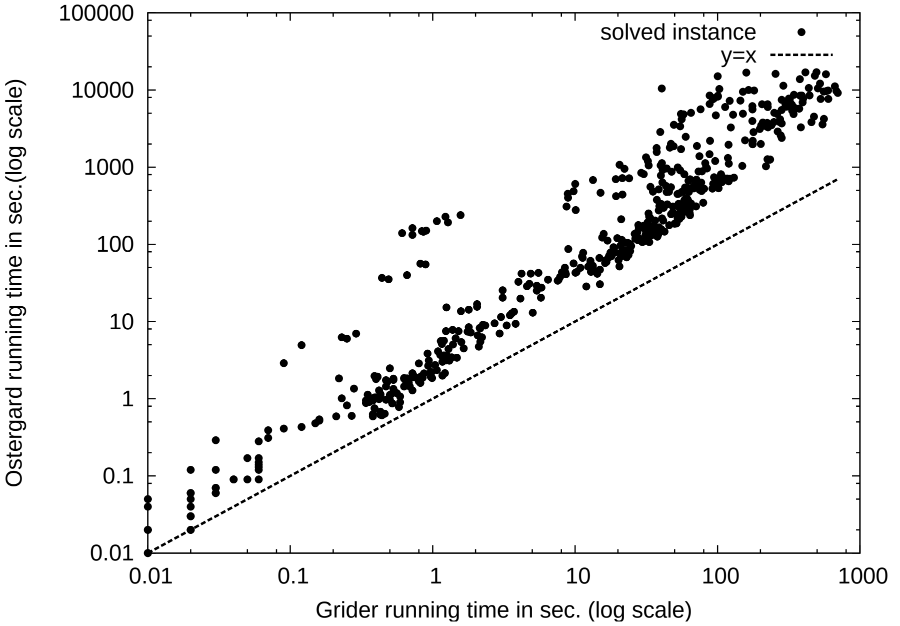
Table 4.6 compares the alignments found by DAST to the ones found by CMO. On similar instances (where both domains come from the same SCOP family), the alignments returned by DAST possess $RMSD_d$ values that are 40% smaller than

Figure 4.5: Running time comparison on the Skolnick set.



ACF versus *Östergård* running time comparison on the set of 164 similar instances from the Skolnick set. The *ACF* time is presented on the x-axis, while the one of *Östergård* is on the y-axis. For all instances except one, *ACF* is faster than *Östergård*.

Figure 4.6: Running time comparison on the Skolnick set.



ACF versus *Östergård* running time comparison on the set of the 509 dissimilar instances from the Skolnick set solved by both algorithms. The *ACF* time is presented on the x-axis, while the one of *Östergård* is on the y-axis. For all instances, *ACF* is faster than *Östergård*.

Table 4.6: Alignment comparison between CMO and DAST.

		Length (AA)		$RMSD_d$ (Å)		Contacts	
	Instance	CMO	DAST	CMO	DAST	CMO	DAST
similar instances	1amkA–1aw2A	**247**	200	1.39	**0.68**	**772**	573
	1amkA–1htiA	**247**	204	1.24	**0.74**	**783**	600
	1qmpA–1qmpB	**129**	118	**0.22**	**0.22**	**389**	383
	1ninA–1plaA	**97**	58	1.42	**0.96**	**295**	131
	1tmhA–1treA	**254**	233	0.90	**0.44**	**862**	768
dissimilar instances	1amkA–1b00A	**120**	41	5.62	**1.23**	**283**	64
	1amkA–1dpsA	**163**	32	13.01	**1.06**	**333**	57
	1b9bA–1dbwA	**123**	44	6.02	**1.11**	**295**	77
	1qmpA–2pltA	**95**	17	7.36	**1.18**	**157**	24
	1rn1A–1b71A	**104**	26	11.22	**0.82**	**196**	49

On ten Skolnick instances, the alignments returned by CMO are compared to the one returned by DAST in terms of length (third and fourth columns), in terms of RMSD of internal distances (fifth and sixth columns) and in terms of number of common contacts (seventh and eighth columns). The similar instances consist in aligning domains coming from the same SCOP family, while the dissimilar instances consist in aligning domains coming from different SCOP families. The alignments returned by DAST always have good $RMSD_d$ values (always less than 2Å), but in order to obtain these good $RMSD_d$ values, the corresponding alignment are shorter than the ones returned by CMO (and notably shorter for dissimilar instances), and also possess less common contacts.

the ones returned by CMO, while their lengths and number of common contacts are about 20% smaller. On dissimilar instances (where the domains come from different families), the alignments returned by DAST always have $RMSD_d$ values smaller than 1.3 Å (though the setting suggested a maximum $RMSD_d$ of 3 Å), which are notably smaller than the $RMSD_d$ values obtained by the CMO's alignments (up to 13 Å). The smaller $RMSD_d$ values of DAST are obtained at the cost of the lengths of the alignments, which are up to 6 times smaller than the ones of CMO.

These results illustrate that DAST succeeds in providing alignment having $RMSD_d$ values smaller than a fixed threshold. However, they also show that the distance threshold of 3 Å was maybe too restrictive, and that using a higher value may be required. Tuning of this parameter needs to be further investigated.

8 Conclusion

In this chapter we introduce a novel protein structure comparison approach DAST, for Distance-based Alignment Search Tool. For any fixed threshold τ, it finds the longest alignment in which each couple of pairs of matched amino-acids shares the same distance relation (+/- τ), and thus the RMSD of the alignment is $\leq \tau$. This property is not guaranteed by the CMO approach, which inspired initially DAST. From a computation standpoint, DAST requires solving the maximum clique problem in a specific k-partite graph. By exploiting the peculiar structure of this graph, we design a new maximum clique solver which significantly outperforms one of the best general maximum clique solver. Our solver was successfully integrated into two protein structure comparison softwares and will be freely available soon. We are currently studying the quality of DAST alignments from a practical viewpoint and comparing the obtained results with other structure comparison methods.

Chapter 5

General conclusion

1 Contributions

In the first chapter of this thesis, we propose a general framework based on alignment graph for modelling protein structure comparison problems. In the second chapter, this framework is applied to the contact map overlap maximisation problem (CMO). Based on the corresponding model, we propose a novel integer programming formulation for CMO which we efficiently solve by designing a dedicated branch and bound algorithm using a Lagrangian relaxation approach. The corresponding solver, A_purva , is compared to the best CMO algorithms from the literature and the obtained results show that A_purva outperforms them both in terms of running time and in terms of quality of the obtained bounds. By using A_purva's results, we are able to obtain automatic classification of protein structure in very good agreements with SCOP. In the third chapter, in order to further accelerate A_purva, we introduce filters based on the secondary structure of the proteins, which allows A_purva to be about 50 times faster. We also propose a hierarchical approach for solving CMO, which consists in first aligning secondary structure elements, and then to use this secondary structure alignment for filtering the amino-acid alignment. Towards this goal we present a new secondary structure alignment approach based on weighted contact maps. This weighted CMO problem is efficiently solved by a modified version of A_purva, and we show that these secondary structure alignments can be used for obtaining a very fast automatic classification of proteins. In the fourth chapter, in order to overcome one of the main weaknesses of CMO, that is, the RMSD of the returned alignment, we propose a new protein structure comparison method based on internal distances, DAST (for Distance-based Alignment Search Tool), whose main characteristic is that the alignments it returns have a RMSD of internal distance smaller than a chosen threshold. DAST is modelled as a maximum clique problems in alignment graphs, and by exploiting the properties of these graphs, we propose a new integer programming formulation for solving maximum cliques problems. In order to solve DAST, we present a new dedicated maximum clique solver that outperforms one of the best maximum clique finder from the literature.

2 Discussions

2.1 Relation between the mathematical models

In CMO, we are looking for a maximum weighted increasing subset of vertices in an alignment graph. Since the edge weights are positive, there is no need to model constraints for activating an edge if its two corresponding endpoint vertices are activated (it will be done in order to maximise the objective function). G. Collet, a Ph.D. student in our team, is currently working on the protein threading problem, which consists in aligning a protein sequence with a protein three dimensional structure. One of the proposed methods, called local [16], can be modelled as a maximum weighted increasing subset of vertices problem in an alignment graph. It is similar to CMO, except that some edge weights can be negative. For this reason, specific constraints for activating the edges associated to negative weights were added into the corresponding integer programming model. DAST is much more constrained in the sense that activating two vertices must activates the edge connecting them. Consequently, the CMO model is the most general (i.e. the less constrained), followed by the local protein threading model, while DAST is the most specific one (i.e. the most constrained).

2.2 Proposed solvers and non-optimality

Both our CMO solver A_purva and our clique solver ACF are branch and bound approaches, but they differ both in terms of branching strategies and in terms of the nature of the bounds used in each sub-problems. A_purva first tries to solve the original problem by iteratively improving the lowerbound (the best feasible solution) and the upperbound (the smallest relaxed solution) found on the full alignment graph. If it can not obtain an optimal solution, then the branch and bound strategy enter in action. If the solving process is stopped before an optimal solution is found, then A_purva can still returns a good feasible solution which corresponds to the full search space. In ACF, the alignment graph is first divided into sub-graphs, according to the ordering of the vertices (the first one being the sub-graph which vertex set contains only the last vertex, the second one is the sub-graph which vertex set contains only the two last vertices, etc.). These problems are then solved from first to last. Unlike A_purva, if the solving process is stopped before an optimal solution is found, then ACF returns a feasible solution which was possibly found in a very small sub-graph. Because of this, ACF can not be used to quickly obtain good feasible solutions.

2.3 About using the secondary structure

We modelled the secondary structure alignment as a one-to-one matching. However, as illustrated in chapter 3, the secondary structure assignments are not perfect. A long secondary structure element can be bent, and secondary structure assignment methods will split it into two or more fragments. This implies that unless the secondary structure assignments get better in terms of quality, the one-to-one matching constraint of the secondary structure alignment should be relaxed. Moreover, the exact beginning and ending of a secondary structure are not well recognised. This needs to be taken into accounts when filtering the amino-acids alignment graph.

3 Future works

First, we are fully aware that, while we worked a lot on the modelling and on the algorithmic aspects of protein structure comparisons, the presented works are missing structural biologist insights for assessing the biological relevance of the methods that we proposed. Second, as illustrated in this thesis, modelling protein structure comparison methods as specific problems in alignment graphs allowed us to propose efficient exact solvers. While our CMO solver A_purva can be used as a heuristic for quickly obtaining good approximated results, this was not its primary objective. Execution times can be a critical point, for example in the case of automatic classification of large databases. We believe that by using the properties of the alignment graphs it is possible to design much faster (and better) heuristics. Third, we used the secondary structure information to reduce de size of the alignment graphs. It may be interesting to use other kinds of informations, which may come from the protein sequence, like the physical properties of the amino-acids (small, polar...), or from the protein three-dimensional structure, like dihedral angles. Fourth, and similarly to what we did for CMO, we would like to propose a hierarchical approach for accelerating DAST. Fith, as in many protein structure comparison methods and as in the original definition of CMO, we supposed that the alignment between two protein structures must be order preserving. By using this order preserving constraint we were able to design efficient solver for the protein structure comparison problem. However, we would like to relax the order preserving constraints in our solvers, in order to apply them on more general problems. Last, we would like to design a langrangian relaxation based approach, similar to the one of A_purva, for solving the maximum clique problem of DAST.

Publication list

Accepted publications

International journals:

- N. Malod-Dognin, R. Andonov and N. Yanev, “Solving Maximum Clique Problem for Protein Structure Similarity”, Serdica Journal of Computing, vol.4, 2010.

International conferences with selection and acts:

- N. Malod-Dognin, R. Andonov and N. Yanev, “Maximum Clique in Protein Structure Comparison”, SEA 2010, the ninth International Symposium on Experimental Algorithms, 2010.
- R. Andonov, N. Yanev and N. Malod-Dognin, “An Efficient Lagrangian Relaxation for the Contact Map Overlap Problem”, K.A. Crandall and J. Lagergren (Eds.) : WABI 2008, LNBI 5251, p. 162-173, 2008.

International conferences with selection but without acts:

- N. Malod-Dognin, R. Andonov and N. Yanev, “Maximum clique approach to protein structure similarity”, MASSEE (mathematical society of south-east europe) International congress on mathematics, Ohrid, Macedonia, p.31, september 16-20,2009.
- N. Malod-Dognin, R. Andonov and N. Yanev, “Maximum clique and similarity measures”, MMSC’09 International Workshop on Mathematical modelling and scientific computations, Velingrad, Bulgaria, p.22, september 23-26,2009.

National conferences with selection and acts:

- N. Malod-Dognin, R. Andonov, N. Yanev and J-F. Gibrat, “Modèle de PLNE pour la recherche de cliques de poids maximal”, ROADEF 2008, p.307-308, 25/02/2008 (In french).

Submitted publications

International journals:

- R. Andonov, N. Yanev and N. Malod-Dognin, “Maximum Contact Map Overlap Revisited”, submitted to the Journal of Computational Biology.

Bibliography

[1] J. Abello, P.M. Pardalos, and M.G.C. Resende. On maximum clique problems in very large graphs. In *In External Memory Algorithms*, pages 119–130. American Mathematical Society, 1999.

[2] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410, 1990.

[3] R. Andonov, S. Balev, and N. Yanev. Protein threading: From mathematical models to parallel implementations. *INFORMS J. on Computing*, 16(4):393–405, 2004.

[4] A. Andreeva, D. Howorth, J-M. Chandonia, S.E. Brenner, T.J.P. Hubbard, C. Chothia, and A.G. Murzin. Data growth and its impact on the scop database: new developments. *Nucl. Acids Res.*, 36:419–425, 11 2007.

[5] E. Balas, S.Ceria, G. Cornuejols, and G. Pataki. Polyhedral methods for the maximum clique problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:11–28, 1996.

[6] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[7] I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo. The maximum clique problem. *Handbook of Combinatorial Optimization.*, 1999.

[8] S.E. Brenner, C. Chothia, and T.J.P. Hubbard. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proceedings of the National Academy of Sciences of the United States of America.*, 95:6073–6078, 1998.

[9] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973.

[10] S. Busygin. A new trust region technique for the maximum weight clique problem. *Discrete Appl. Math.*, 154(15):2080–2096, 2006.

[11] A. Caprara, R. Carr, S. Israil, G. Lancia, and B. Walenz. 1001 optimal pdb structure alignments: integer programming methods for finding the maximum contact map overlap. *J. Comput. Biol.*, 11(1):27–52, 2004.

[12] A. Caprara and G. Lancia. Structural alignment of large—size proteins via lagrangian relaxation. In *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*, pages 100–108, New York, NY, USA, 2002. ACM.

[13] M. Carpentier, S. Brouillet, and J. Pothier. Yakusa: a fast structural database scanning method. *Proteins.*, 61:137–151, 10 2005.

[14] R. Carr, G. Lancia, and S. Istrail. Branch-and-cut algorithms for independent set problems: Integrality gap and an application to protein structure alignment. Technical report, Sandia National Laboratories, 2000.

[15] F. Cazals and C. Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1-3):564 – 568, 2008.

[16] G. Collet, R. Andonov, N. Yanev, and J-F. Gibrat. Local Protein Threading by Mixed Integer Programming. Research report n^o 7122, 2009.

[17] A.K. Dunker, J.D. Lawson, C.J. Brown, R.M. Williams, P. Romero, J.S. Oh, C.J. Oldfield, A.M. Campen, C.M. Ratliff, K.W. Hipps, J. Ausio, M.S. Nissen, R. Reeves, C. Kang, C.R. Kissinger, R.W. Bailey, M.D. Griswold, W. Chiu, E.C. Garner, and Z. Obradovic. Intrinsically disordered protein. *Journal of Molecular Graphics and Modelling*, 19(1):26 – 59, 2001.

[18] Alexis Falicov and Fred E. Cohen. A surface of minimum area metric for the structural comparison of proteins. *Journal of Molecular Biology*, 258(5):871–892, 1996.

[19] M.L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics.*, 11:29–35, 1 1975.

[20] D. Frishman and P. Argos. Knowledge-based protein secondary structure assignment. *PROTEINS: Structure, Function, and Genetics.*, 23:566–579, 1995.

[21] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[22] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study.*, 2:82–114, 1974.

[23] S. German and D. German. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 6(6):721–741, 1984.

[24] M. Gerstein and M. Levitt. Using iterative dynamic programming to obtain accurate pair-wise and multiple alignments of protein structures. *ISMB-96 Proceedings*, pages 59–67, 1996.

[25] M. Gerstein and M. Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Protein Science.*, 7:445–456, 1998.

[26] J-F. Gibrat, T. Madej, and S.H. Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology.*, 6:377–385, 06 1996.

[27] A. Godzik. The structural alignment between two proteins: Is there a unique answer? *Protein Science*, 5(7):1325–1338, 1996.

[28] A. Godzik and J. Skolnick. Flexible algorithm for direct multiple alignment of protein structures and seequences. *CABIOS*, 10:587–596, 1994.

[29] D. Goldman, S. Israil, and C. Papadimitriu. Algorithmic aspects of protein structure similarity. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 512–521, Washington, DC, USA, 1999. IEEE Computer Society.

[30] M. Guignard. Lagrangean relaxation. *TOP*, 11(2):151–200, 01 2003.

[31] A. Harrison, F. Pearl, I. Sillitoe, T. Slidel, R. Mott, J. Thornton, and C. Orengo. Recognizing the fold of a protein structure. *Bioinformatics*, 14(19):1748–1759, 09 2003.

[32] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, (18):1138–1162, 1970.

[33] M. Held, P. Wolfe, and H.P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974.

[34] J.H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992.

[35] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology.*, 223:123–138, 1993.

[36] L. Holm and C. Sander. The fssp database of structurally aligned protein fold families. *Nucleic Acids Research*, 22(17):3600–3609, 1994.

[37] L. Holm and C. Sander. Mapping the protein universe. *Science*, 273:595–602, 1996.

[38] B.J. Jain and M. Lappe. Joining softassign and dynamic programming for the contact map overlap problem. In Sepp Hochreiter and Roland Wagner, editors, *BIRD*, volume 4414 of *Lecture Notes in Computer Science*, pages 410–423. Springer, 2007.

[39] W. Kabsch and C. Sander. Dictionary of protein secondary structure: Patter recognition of hydrogen-bonded and geometrical features. *Biopolymers.*, 22:2577–2637, 1983.

[40] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations.*, 6:85–103, 06 1972.

[41] G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal pdb structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *RECOMB '01: Proceedings of the fifth annual international conference on Computational biology*, pages 193–202, 2001.

[42] R.H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is np-complete. *Protein Engineering.*, 7(9):1059–1068, 1994.

[43] I.C. Lerman. Likelihood linkage analysis (lla) classification method (around an example treated by hand). *Biochimie*, 75(5):379–397, 1993.

[44] I.C. Lerman, P. Peter, and H. Leredde. Principles and calculus of the implanted method in the chavl program (hierarchical classification by the analysis of the likelihood of the bonds) (in french). *La Revue de Modulad*, 13:63–90, 1994.

[45] J. Martin, G. Letellier, A. Marin, J-F. Taly, A.G. de Brevern, and J-F. Gibrat. Protein secondary structure assignment revisited: a detailed analysis of different assignment methods. *BMC Structural Biology.*, 5:17, 2005.

[46] A.C.W. May and M.S. Johnson. Protein structure comparisons using a combination of a genetic algorithm, dynamic programming and least-squares minimization. *Protein Eng.*, 7(4):475–485, 1994.

[47] J. McGregor. Backtrack search algorithms and the maximal common subgraph. *Software Practice abd Experience*, 12:23–34, 1982.

[48] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association.*, 44(247):335–341, 1949.

[49] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology.*, 247:536–540, 1995.

[50] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.

[51] C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. Cath - a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[52] P.R.J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Appl. Math.*, 120(1-3):197–207, 2002.

[53] P.M. Pardalos and G.P. Rodgers. A branch and bound algorithm for the maximum clique problem. *Comput. Oper. Res.*, 19(5):363–375, 1992.

[54] D. Pelta, J. Gonzales, and M. Vega. A simple and fast heuristic for protein structure comparison. *BMC Bioinformatics*, 9(1):161, 2008.

[55] G.A. Petsko and D. Ringe. *Protein Structure and Function: Primers in Biology.* New Sciences Press Ltd, Blackwell Publishing Ltd and Sinauer Associates,Inc. publishers., 2004.

[56] W. Pullan. Protein structure alignment using maximum cliques and local search. *AI 2007: Advances in Artificial Intelligence.*, pages 776–780, 2007.

[57] Oliver C Redfern, Andrew Harrison, Tim Dallman, Frances M. G Pearl, and Christine A Orengo. Cathedral: A fast and effective algorithm to predict folds and domain boundaries from multidomain protein structures. *PLoS Computational Biology*, 3(11):2333–2347, 11 2007.

[58] J-C. Régin. Using constraint programming to solve the maximum clique problem. *CP 2003 : principles and practice of constraint programming, Lecture notes in computer science.*, 2833:634–648, 2003.

[59] M.L. Sierk and G.J. Kleywegt. Déjà vu all over again: Finding and analyzing protein structure similarities. *Structure*, 12(12):2103–2111, 2004.

[60] A.P. Singh and D.L. Brutlag. Protein structure alignment: A comparison of methods., 2000.

[61] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[62] R. Sowdhamini and T.L. Blundell. An automatic method involving cluster analysis of secondary structures for the identification of domains in proteins. *Protein Science*, 4(3):506–520, 1995.

[63] D.M. Strickland, E. Barnes, and J.S. Sokol. Optimal protein structure alignment using maximum cliques. *Oper. Res.*, 53(3):389–402, 2005.

[64] S. Subbiah, D. V. Laurents, and M. Levitt. Structural similarity of dna-binding domains of bacteriophage repressors and the globin core. *Current Biology*, 3(3):141–148, 1993.

[65] W.R.E. Taylor. The classification of amino acid conservation. *Journal of Theoretical Biology.*, 119(2):205–218, 1986.

[66] I. Wohlers, L. Petzold, F.S. Domingues, and G. Klau. Aligning protein structures using distance matrices and combinatorial optimization. *Lecture Notes in Informatics: Proceedings of GCB 2009 (German Conference on Bioinformatics)*, 2009.

[67] I. Wohlers, L. Petzold, F.S. Domingues, and G. Klau. Paul: protein structural alignment using integer linear programming and lagrangian relaxation. *BMC Bioinformatics*, 10(Suppl 13):P2, 2009.

[68] J. Xiang and M. Hu. Comparisons of protein structure alignment methods: Rigid and flexible, sequential and non-sequential. In *Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on*, pages 21–24, May 2008.

[69] W. Xie and N. Sahinidis. A reduction-based exact algorithm for the contact map overlap problem. *Journal of Computational Biology*, 14(5):637–654, 2007.

[70] N. Yanev, P. Veber, R. Andonov, and S. Balev. Lagrangian approaches for a class of matching problems in computational biology. *Comput. Math. Appl.*, 55(5):1054–1067, 2008.

[71] J. Zhu and Z. Weng. Fast: a novel protein structure alignment algorithm. *Proteins*, 58(3):618–627, February 2005.

Abstract

In structural biology, it is commonly admitted that the three dimensional structure of a protein determines its function. A fruitful assumption based on this paradigm is that proteins sharing close three dimensional structures may derive from the same ancestor and thus, may share similar functions. Computing the similarity between two protein structures is therefore a crucial task and has been extensively investigated. Among all the proposed methods, we focus on the similarity measure called Contact Map Overlap maximisation (CMO), mainly because it provides scores which can be used for obtaining good automatic classifications of the protein structures.

In this thesis, comparing two protein structures is modelled as finding specific sub-graphs in specific k-partite graphs called alignment graphs, and we show that this task can be efficiently done by using advanced combinatorial optimisation techniques. In the first part of the thesis, we model CMO as a kind of maximum edge induced sub-graph problem in alignment graphs, for which we conceive an exact solver which outperforms the other CMO algorithms from the literature. Even though we succeeded to accelerate CMO, the procedure still stays too much time consuming for large database comparisons. The second part of the thesis is dedicated to further accelerate CMO by using structural biology knowledge. We propose a hierarchical approach for CMO which is based on the secondary structure of the proteins. Finally, although CMO is a very good scoring scheme, the alignments it provides frequently posses big root mean square deviation values. To overcome this weakness, in the last part of the thesis, we propose a new comparison method based on internal distances which we call DAST (for Distance-based Alignment Search Tool). It is modelled as a maximum clique problem in alignment graphs, for which we design a dedicated solver with very good performances.

Keywords : protein structure comparison, contact map overlap maximisation, k-partite graph, maximum edge induced sub-graph, maximum clique, integer programming, branch and bound.

Résumé

En biologie structurale, il est couramment admit que la structure tridimensionnelle d'une protéine détermine sa fonction. Ce paradigme permet de supposer que deux protéines possédant des structures tridimensionnelles similaires peuvent partager un ancêtre commun et donc posséder des fonctions similaires. Déterminer la similarité entre deux structures de protéines est une tâche importante qui a été largement étudiée. Parmi toutes les méthodes proposées, nous nous intéressons à la mesure de similarité appelée "maximisation du recouvrement de cartes de contacts" (ou CMO), principalement parce qu'elle fournit des scores de similarité pouvant être utilisés pour obtenir de bonnes classifications automatiques des structures de protéines.

Dans cette thèse, la comparaison de deux structures de protéines est modélisée comme une recherche de sous-graphe dans des graphes k-partis spécifiques appelés graphes d'alignements, et nous montrons que cette tâche peut être efficacement réalisée en utilisant des techniques avancées issues de l'optimisation combinatoire. Dans la seconde partie de cette thèse, nous modélisons CMO comme une recherche de sous-graphe maximum induit par les arêtes dans des graphes d'alignements, problème pour lequel nous proposons un solveur exact qui surpasse les autres algorithmes de la littérature. Même si nous avons réussi à accélérer CMO, la procédure d'alignement requière encore trop de temps de calculs pour envisager des comparaisons à grande échelle. La troisième partie de cette thèse est consacrée à l'accélération de CMO en utilisant des connaissances issues de la biologie structurale. Nous proposons une approche hiérarchique pour résoudre CMO qui est basée sur les structures secondaires des protéines. Enfin, bien que CMO soit une très bonne mesure de similarité, les alignements qu'elle fournit possèdent souvent de fortes valeurs de déviation (root mean squared deviation, ou RMSD). Pour palier à cette faiblesse, dans la dernière partie de cette thèse, nous proposons une nouvelle méthode de comparaison de structures de protéines basée sur les distances internes que nous appelons DAST (pour Distance-based Alignment Search Tool). Elle est modélisée comme une recherche de clique maximum dans des graphes d'alignements, pour laquelle nous présentons un solveur dédié montrant de très bonnes performances.

Mots clés : comparaison de structures de protéines, recouvrement de cartes de contacts, graphes *k*-partis, clique maximum, programmation linéaire en nombres entiers, algorithmes par séparation et évaluation.

VU :	VU :
Le Directeur de Thèse (Nom et prénom)	**Le Responsable de l'École Doctorale** (Nom et Prénom)

VU pour autorisation de soutenance

Rennes, le

Le Président de l'Université de Rennes 1

Guy CATHELINEAU

VU après soutenance pour autorisation de publication :

Le Président de Jury,
(Nom et Prénom)