Nouveaux logiciels pour la biologie structurale computationnelle et la chémoinformatique

François Bérenger

▶ **To cite this version:**

François Bérenger. Nouveaux logiciels pour la biologie structurale computationnelle et la chémoinformatique. Chemo-informatique. Conservatoire national des arts et metiers - CNAM, 2016. Français. NNT : 2016CNAM1047 . tel-01416562

HAL Id: tel-01416562
https://theses.hal.science/tel-01416562

Submitted on 14 Dec 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

le cnam

École Doctorale Sciences des Métiers de l'Ingénieur

Laboratoire Génomique Bioinformatique et Applications

Thèse de doctorat

présentée par : **François Bérenger**

soutenue le : **5 juillet 2016**

pour obtenir le grade de : **Docteur du Conservatoire National des Arts et Métiers**

Discipline / Spécialité : **Bioinformatique**

Nouveaux logiciels pour la biologie structurale computationnelle et la chémoinformatique

Directeur

M. Zagury Jean-François	*Professeur, CNAM, Paris*

Rapporteurs

M. Horvath Dragos	*Directeur de recherche, CNRS, Strasbourg*
M. Tufféry Pierre	*Directeur de recherche, INSERM, Paris*

Éxaminateur

M. Molina Christophe	*Président directeur général, société PIKAÏROS, Toulouse*

Président du Jury

M. Port Marc	*Professeur, CNAM, Paris*

Table des matières

Table des figures

Chapitre 1

Remerciements

Je remercie ma mère, à qui je dois beaucoup. Pour ajouter à ma dette, je dois aussi à l'ancienne maîtresse d'école la correction orthographique et grammaticale des parties en langue française de ce document.

Je remercie aussi mes anciens collègues du RIKEN : les bonnes idées présentées ici sont souvent celles qui ont résisté à une discussion acharnée avec eux ou qui ont été améliorées après plusieurs échanges. Les mauvaises idées, ou celles avec peu de chance de succès, ne durent pas longtemps lors d'une discussion avec d'autres scientifiques. Je pense tout particulièrement aux docteurs Yong Zhou, David Simoncini et Arnout Voet.

Je remercie aussi Kam Zhang, pour m'avoir recruté au RIKEN et avoir autorisé l'effort en logiciels libres dans notre équipe de recherche. Je remercie aussi Xavier Rival de l'INRIA pour l'effort en logiciels libres dans l'équipe Antique ainsi que le temps que j'ai pu passer sur ma thèse ou à des conférences.

Je remercie le professeur Jean-François Zagury pour la direction de cette thèse. Je remercie les rapporteurs Dragos Horvath et Pierre Tufféry ainsi que l'examinateur Christophe Molina pour leur travail.

Je remercie Maxime Viarouge pour les logos des logiciels PAR et Durandal.

Enfin, je remercie ma femme Yukari ainsi que mes deux enfants Mamolou et Gauvin pour me faire rire et me rappeler chaque jour ce qui est important dans la vie.

Chapitre 2

Introduction et état de l'art

De août 2009 à juin 2014, j'ai travaillé au Japon dans le campus du RIKEN de Wakoushi, dans une équipe de bioinformatique structurale et de découverte de médicament par ordinateur. L'équipe existe toujours et est dirigée par Kam Y. J. Zhang. J'ai été embauché en tant que « research associate ». Un intitulé de poste qui n'existe pas en France et qu'on pourrait traduire par chercheur associé. J'y ai observé pour la première fois les structures 3D de protéines et de petites molécules sur des écrans d'ordinateur. Certains des sujets de l'équipe m'ont vivement intéressé. Au cours de mon travail, j'ai eu accès au supercalculateur du RIKEN nommé RICC, qui comporte pas moins de 8000 processeurs pour le calcul scientifique.

Mes travaux scientifiques publiés ont porté sur quatre sujets distincts. En bioinformatique structurale, j'ai accéléré un algorithme de regroupement largement utilisé en prédiction de structures de protéines et en analyse de trajectoires de dynamique moléculaire. J'ai aussi créé une méthode pour comparer dans l'espace électrostatique une petite molécule avec la protéine qu'elle est censée remplacer au sein d'une interface protéine-protéine. En chémoinformatique, j'ai montré comment utiliser efficacement la fonction d'autocorrélation afin d'encoder de façon rotation-translation invariante une petite molécule en 3D dans les trois espaces chimiques majeures (stérique, hydrophobe et électrostatique). Cette encodage très fin de la molécule est non seulement rapide mais il a aussi montré de très bonnes performances dans des tâches de criblage d'une chimiothèque virtuelle. Enfin, dans le domaine du calcul haute performance, j'ai créé des outils transversaux qui permettent

d'accélérer la préparation et l'exécution d'expériences computationnelles.

Cette thèse est écrite dans l'optique de pouvoir être lue par tout lecteur ayant suivi un cursus scientifique universitaire. Sa lecture ne devrait pas exiger d'être spécialiste en informatique, en biologie structurale ou en chimie. De plus, étant partisan du principe de parcimonie, nous nous sommes intéressés à des idées simples. Dans les annexes de ce document se trouve un glossaire où tous les sigles rencontrés sont expliqués.

Dans une optique de reproductibilité des résultats, nos logiciels ont tous été publiés sous des licences pour le logiciel libre. Quand nous avons dû créer un jeu de données pour valider scientifiquement un logiciel, ce jeu de données à lui aussi été mis à disposition du public de façon libre et gratuite. Toute recherche scientifique se doit d'être reproductible. Nous pensons que la libre disponibilité des articles de recherche, des logiciels ainsi que des jeux de données permet d'accélérer la recherche, la dissémination des connaissances ainsi que leur adoption par le plus grand nombre.

Cette thèse se découpe en trois grandes parties. Dans la première partie nous introduisons les concepts utiles à la compréhension des résultats publiés. Nous listons et résumons ensuite ces résultats dans la deuxième partie. Dans la troisième partie, nous discutons ces résultats. Nous y dévoilons aussi des résultats non publiés ainsi que quelques pistes pour aller plus loin avec certaines des idées que nous avons explorées.

Dans la section qui suit, nous introduisons brièvement la bioinformatique ainsi que ses sous domaines, pour situer le cadre dans lequel s'inscrivent nos travaux.

2.1 La bioinformatique

La bioinformatique est une science multidisciplinaire. Elle se situe au carrefour entre (au moins) la biologie, l'informatique ainsi que les mathématiques [102]. Au sens large, la bioinformatique vise à stocker, indexer, analyser, modéliser et prédire à partir d'informations provenant de données biologiques expérimentales. Son but ultime est de permettre la compréhension de phénomènes biologiques. Dû au récent déluge de données génétiques, telles que celles produites par le projet de séquençage du génome humain [90]), les biologistes sont dépendants des ordinateurs et des bases de données pour de nombreuses tâches.

On peut découper les buts de la bioinformatique en trois axes :

1. le stockage, l'indexation et la restitution de données
2. le développement d'outils qui permettent d'analyser ces données
3. l'utilisation de ces outils et l'interprétation de leurs résultats afin d'apporter de nouvelles connaissances en biologie.

En fonction des données considérées, on peut découper la bioinformatique en sous domaines. Pour chaque domaine, on donne un ordre de grandeur de la taille des données manipulées.

1. L'analyse de séquences d'ADN. On travaille ici sur des séquences de nucléotides (Adénine, Cytosine, Guanine ou Thymine). Un chromosome humain contient en moyenne 129M de paires de bases et encode 800 protéines [43].
2. L'analyse de séquences de protéines. On travaille ici sur des séquences d'acides aminés (abrégé AA ; il existe 20 acides aminés). Une protéine humaine fait en moyenne 338 AA de long [26].
3. L'analyse de structures macromoléculaires. On travaille alors sur une représentation à l'échelle atomique et en 3D d'une protéine, d'une molécule d'ADN ou d'ARN. Si l'on fait l'approximation de 19 atomes par AA [93], une seule protéine humaine contient alors 6400 atomes en moyenne.
4. L'analyse de génomes. On travaille alors sur des génomes entiers. Le génome humain fait une longueur de trois milliards de paires de bases [43]. Comparer des génomes permet d'établir des arbres phylogénétiques, qui illustrent la proximité génétique entre espèces.
5. L'expression des gènes. On cherche à décoder quelle protéine ou quel ARN est encodé par quel gène. Un gène humain a une longueur moyenne de 28000 paires de bases [166].

La majorité de nos travaux sont des outils qui travaillent à partir d'information structurale. C'est à dire que le modèle de protéine avec lequel nous travaillons est en 3D et à l'échelle atomique. On parle alors de biologie structurale computationnelle ou de bioinformatique structurale. Dès lors qu'une structure est disponible, on peut faire de la

modélisation moléculaire, essayer de positionner une molécule au sein d'une protéine (docking) et analyser des complexes protéine-ligand ou protéine-protéine par exemple (ligand signifie petite molécule). Il peut aussi s'agir d'une autre protéine ou d'un peptide (une protéine très courte).

D'ailleurs, un des défis majeurs de la bioinformatique structurale est de prédire la forme d'une protéine en fonction de sa séquence d'acides aminés.

2.2 La prédiction de structures de protéines

FIGURE 2.1 – Myoglobine de baleine (un transporteur d'oxygène) : première protéine dont la structure à été résolue par cristallographie. Illustration de David Goodsell pour l'article de la protéine du mois d'octobre 2011 sur le site `www.pdb.org` (image sous licence creative commons).

Le pliage de protéine vise à prédire la structure tridimensionnelle d'une protéine (figure 2.1) en connaissant seulement sa séquence d'acides aminés (AA).

Le principe de base est le postulat d'Anfinsen (prix Nobel 1972) : la forme d'une protéine est déterminée seulement par sa séquence d'acides aminés[5]. Pour une protéine relativement petite et de forme globulaire, la conformation stable de la protéine est proche du minimum global de sa fonction d'énergie : ses acides aminés hydrophobes se trouvent au centre de la conformation alors que ceux hydrophiles sont exposés au solvant (l'eau) à la surface.

Le pliage de protéines est une tâche difficile car la fonction d'énergie utilisée est empirique, entachée d'erreurs et le problème possède une grande combinatoire[96]. Concrète-

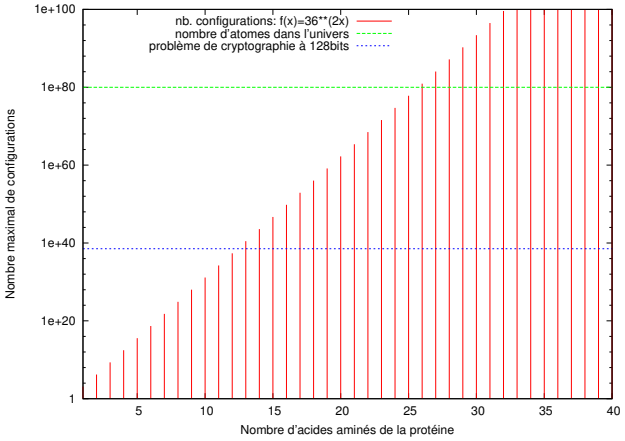


FIGURE 2.2 – Combinatoire du problème de prédiction de la structure tridimensionnelle d'une protéine. On considère un pas de dix degrés pour chaque axe de rotation et on omet le fait que certaines configurations sont inatteignables à cause des collisions entre atomes. Pour donner des ordres de grandeur, on montre aussi deux grands nombres via deux lignes horizontales : le nombre d'atomes de l'univers et le nombre de clés de longueur 128 bits possibles. Les clés 128 bits sont considérées incassables par les cryptanalystes lors d'une attaque par énumération exhaustive.

ment, chaque acide aminé a deux degrés de liberté au niveau du squelette peptidique. Si l'on considère seulement des rotations par pas de 10 degrés pour chaque degré de liberté, on obtient un nombre de configurations possibles qui grossit extrêmement vite en fonction du nombre d'acides aminés (figure 2.2). C'est le paradoxe de Levinthal [95] : explorer séquentiellement toutes les conformations possibles dans l'espoir de trouver celle de plus basse énergie est calculatoirement impossible alors que les protéines adoptent leur conformation native en quelques secondes tout au plus [99,133]. Ce qui suggère que les protéines ne se plient pas de façon aléatoire mais suivent une « trajectoire » menant vers la conformation de plus basse énergie.

Il existe plusieurs approches qui ont fait leurs preuves en prédiction de structure, tels les logiciels TASSER [184,186] et ROSETTA [119]. Pour une amélioration de l'algorithme de recherche de ROSETTA afin de biaiser la recherche vers l'utilisation de fragments de bonne qualité, on peut se rapporter au logiciel EdaFold [141]. PEP-FOLD est lui un logiciel spécialisé dans la prédiction de structures de peptides ou de mini protéines [137]. L'état de l'art est évalué tous les deux ans lors de l'expérience CASP [110].

Le pliage de protéine *in-silico* est une simulation. Mais pour avoir la structure réelle d'une protéine on a recours à des techniques expérimentales telles que la cristallographie [83] ou la résonance magnétique nucléaire [177] pour arriver à « photographier » une protéine avec une précision à l'échelle atomique.

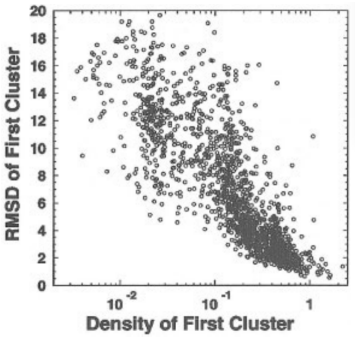


FIGURE 2.3 – RMSD (au centre du plus gros cluster) à la protéine native dont on à essayé de reproduire la forme, en fonction de la densité du plus gros cluster. Figure reproduite avec l'autorisation de l'éditeur.

L'utilité du regroupement dans l'analyse des résultats de simulations de pliage de protéines à été montrée par Yang Zhang et Jeffrey Skolnick dans leur publication sur le logiciel SPICKER [185]. La figure clé de leur article (reproduite ici en figure 2.3) montre que les modèles de meilleure qualité se trouvent en général dans les clusters de plus haute densité. Les clusters du logiciel SPICKER sont obtenus par l'algorithme "exact clustering" (décrit plus loin). Tous les clusters ont le même diamètre (un paramètre de l'algorithme) et le nombre de conformations dans un cluster est donc une mesure de sa densité.

2.2.1 La conception de protéines

La conception de protéines (« protein design » en anglais) est le problème inverse de la prédiction de structure de protéines [2,28,55,142,159]. Le problème consiste à calculer une séquence d'acides aminés qui se pliera en la forme voulue. La complexité du problème est de 20^N, N étant la longueur de la séquence d'acides aminés considérée. À partir de seulement 30 acides aminés, le problème est plus dur que le problème cryptographique qui consiste à casser par force brute une clé de taille 128 bits.

Malgré la difficulté du problème, on trouve de nombreux succès récents. Ceci s'explique

FIGURE 2.4 – La protéine PDB:3B5L est une enzyme artificielle (rétroaldolase) issue de travaux en conception de protéine.

sûrement par le fait que s'il y a un intérêt théorique à comprendre comment une séquence d'acides aminés se plie ; il y a un intérêt pratique à résoudre le problème inverse : on peut alors concevoir des protéines qui ont une fonctionnalité ou une forme voulue. Par exemple, créer de nouvelles enzymes (figure 2.4) qui n'existent pas dans la nature et qui permettent de catalyser certaines réactions chimiques[77,101]. La protéine top7[86], créée en 2003, présente une forme non présente dans la nature et est le fruit d'une conception assistée par ordinateur (figure 2.5).

Parmi quelques logiciels connus en protein design, on peut citer Rosetta design ainsi que OSPREY[47]. Les utilisateurs qui souhaitent aider et fournir de la puissance de calcul aux travaux de David Baker ainsi que ses collègues peuvent installer le logiciel Rosetta@home[31]. Le logiciel Foldit est lui aussi d'intérêt puisqu'il permet de façon ludique d'essayer de plier une séquence d'acide aminés donnée en une forme globulaire sous la forme d'un petit jeu vidéo 3D utilisant la souris pour naviguer autour de la protéine mais permettant aussi de saisir et de bouger certaines parties.

Certains de nos travaux de recherche ont donné lieu à des outils qui travaillent sur des petites molécules (on parle aussi de ligands). Dans la partie qui suit nous introduisons donc le domaine de la chémoinformatique.

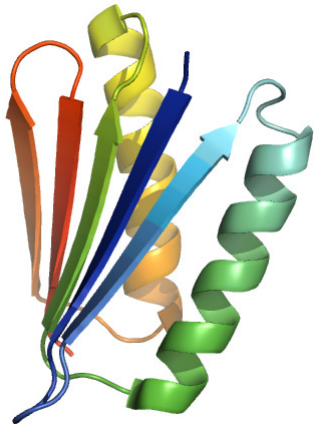


FIGURE 2.5 – La protéine « top7 » (PDB:1QYS) est une protéine de 93 acides aminés créée par l'homme et de forme inconnue dans la nature.

2.3 La chémoinformatique

Pour une bonne introduction à la chémoinformatique, on peut se rapporter à deux articles[20,39] et à au moins un livre[48].

La chémoinformatique est la science qui vise à stocker, manipuler et analyser de l'information chimique. On confond parfois la chémoinformatique avec la chimie computationnelle, mais cette dernière est plutôt une branche théorique de la chimie. La chémoinformatique est un peu à la chimie ce que la bioinformatique est à la biologie. La chémoinformatique vise à créer des outils informatiques utiles au chimiste. D'après Johann Gasteiger, l'un des pères fondateurs du domaine : « la chémoinformatique consiste à appliquer des méthodes informatiques pour résoudre des problèmes de chimie ».

Listons ici quelques tâches du domaine de la chémoinformatique.

1. Représenter, stocker et visualiser des molécules. De la représentation la plus simple à la plus complète, on peut représenter une molécule par son nom, sa formule chimique, son graphe de connectivité en 2D ou sa structure en 3D avec visualisation de la surface.
2. Représenter des réactions chimiques et les stocker dans des bases de données de réactions telles que le « Chemical Abstracts Service[36] ».

3. Générer des conformations 3D de molécules[51,65].
4. Créer des méthodes de recherche de molécules (recherche complète, recherche de sous-structure, recherche Markush ou recherche par similarité).

Ayant développé une méthode de recherche moléculaire par calcul de similarité, nous détaillons un peu plus ce concept dans ce qui suit.

2.3.1 Encodage de molécules et recherche par similarité

La similarité moléculaire est un sujet bien exploré de la chemoinformatique[9,12,13,173] et de la chimie thérapeutique[85]. La similarité moléculaire est utilisée entre autres en LBVS[38,70,123,176] et pour prédire les effets secondaires d'une molécule.

Un encodage classique de molécule est l'empreinte qui consiste en la conversion de la molécule en un vecteur de booléens. Par exemple, la très populaire empreinte MACCS consiste en un tel encodage. Il existe cependant une infinité d'autres encodages possibles, avec des degrés de précision différents. Certains encodages préservent mieux l'information contenue dans la molécule de départ que l'empreinte MACCS qui considère juste la molécule de départ comme un objet 2D. On parle ici d'encodage avec perte[130], comme dans le cas des images JPEG, de la musique au format MP3 ou plus généralement en mathématiques de ce qu'on appelle les fonctions de hachage. La perte d'information implique qu'il est difficile de revenir en arrière étant donné la seule empreinte. Si l'on revient en arrière, il risque alors d'y avoir une ambiguïté en ce qui concerne la molécule de départ (il existe plusieurs molécules possibles aboutissant à cette empreinte). En particulier, certains encodages peuvent encoder une conformation 3D de la molécule considérée. En pratique, une empreinte de molécule se compose d'une représentation (par exemple un vecteur de nombre réels), optionnellement combinée à des poids. Mais pour faire un calcul de similarité il faut aussi disposer d'une fonction de score (tel que le score de Jaccard/Tanimoto) ou d'une distance définie sur l'espace des empreintes.

Cette possibilité de comparer des empreintes de molécules ouvre la porte à l'analyse par regroupement et au criblage virtuel basé sur des ligands (LBVS). Le choix de la représentation, des éventuels poids associés ainsi que de la fonction de score est empirique et dépend de l'application envisagée. Ils doivent être choisis afin d'être maximalement utiles

au chimiste. Il est aussi avantageux que la combinaison choisie soit rapide, étant donné que de nos jours une chimiothèque virtuelle peut contenir un nombre colossal de molécules. Pour donner un ordre de grandeur, il y a 1700 millions de molécules non redondantes (avec leur route de synthèse) dans la chimiothèque virtuelle du « RIKEN Quantitative Biology Center »[63]. La recherche par similarité est si utile en chémoinformatique et chimie médicinale à cause du principe de similarité. En effet, des molécules similaires exhibent souvent la même activité biologique et les même propriétés physico-chimiques. La recherche par similarité est utilisée fréquemment lors des premières phases d'un projet de découverte de molécule active. Car, à ce moment du projet, seul peu d'information structure-activité est disponible. En dépit d'être un concept simple, la recherche de molécules par calcul de similarité est très puissante.

La chémoinformatique est un domaine d'application de l'informatique passionnant. On ne fait plus de l'informatique pour les informaticiens, mais de l'informatique pour les chimistes. Les logiciels du domaine manipulent des molécules, que l'on peut visualiser et certaines de ces molécules peuvent être synthétisées ou sont présentes naturellement dans le monde réel. L'utilisateur d'un logiciel du domaine est un chimiste et c'est pour lui au final que l'outil doit être utile (et si possible rapide). De manière amusante, on peut relier bioinformatique, chémoinformatique et chimie quantique. La bioinformatique est la science qui traite les plus grosses molécules alors que la chémoinformatique traite des molécules plus petites et que la chimie quantique va entrer à l'échelle subatomique dans le détail de ces petite molécules. Il est parfois difficile de déterminer dans laquelle de ces sciences on travaille exactement. Par exemple, faire du docking revient à travailler à l'interface entre bioinformatique structurale et chémoinformatique car on étudie les interactions entre une petite molécule au sein d'un protéine (qui est elle une macromolécule).

Les ligands et les protéines peuvent être encodés sous la forme de vecteurs dans un espace à N dimensions. On parle alors de descripteur moléculaire.

2.4 Les descripteurs moléculaires

Les descripteurs moléculaires sont une notion très importante en chémoinformatique. D'après Roberto Todeschini et Viviana Consonni, un descripteur moléculaire est le résultat final d'une opération mathématique et logique qui transforme de l'information chimique en un nombre utile ou le résultat d'une expérience standard.

Il existe trois classes de descripteurs. Les descripteur 1D que l'on peut calculer à partir de la composition de la molécule. Par exemple la masse moléculaire, le nombre d'atomes lourds, etc. Les descripteurs 2D que l'on peut calculer à partir du graphe moléculaire, comme les signatures MACCS et FP4. Enfin, les descripteurs 3D qui sont calculés à partir d'une ou plusieurs conformations moléculaires, comme la surface polaire ou la surface accessible au solvant.

Pour une référence encyclopédique sur les descripteurs moléculaires, on pourra se rapporter au livre en deux volumes « Molecular descriptors for chemoinformatics »[157]. Une très grande quantité de descripteurs est implémentée dans le logiciel DRAGON[105]. Pour les chercheurs qui peuvent se permettre de dévoiler leurs molécules sur Internet, il existe un service gratuit en ligne appelé E-Dragon[153] [1]. Sinon, il existe une licence DRAGON permanente a prix raisonnable pour les chercheurs du monde académique. L'excellente librairie libre Open Babel[116] permet elle aussi de calculer quelques descripteurs moléculaires courants.

Les descripteurs moléculaires sont très utilisés pour quantifier la similarité moléculaire[152] et créer des modèles reliant l'activité d'une molécule à sa structure (QSAR). Pour le calcul rapide du sous graphe commun maximal entre deux molécules, il existe de nombreuses méthodes[58,81,122] approximatives.

2.4.1 Le modèle de Wildman et Crippen

Nous présentons ici un descripteur moléculaire qui permet de travailler avec une molécule dans l'espace lipophile (équivalent d'hydrophobe). Ce descripteur est utilisé par le logiciel ACPC, décrit à la section 3.4.

1. `http://www.vcclab.org`

Définition :

$$LogP = \log(\frac{C_o}{C_e}) \tag{2.1}$$

avec C_o la concentration du composé dissous dans l'octanol et C_e la concentration du composé dissous dans l'eau.

Le LogP est une valeur mesurée en laboratoire qui permet de déterminer si un composé chimique est hydrophobe ou hydrophile, l'octanol et l'eau étant des solvants non miscibles. Si le composé chimique est plus présent dans l'octanol à la fin de l'expérience, LogP est positif et le composé est jugé hydrophobe. Dans le cas contraire, LogP est négatif et le composé est jugé hydrophile.

En 1999, Scott Wildman et Gordon Crippen publient un modèle qui assigne à chaque atome d'une molécule une contribution à la valeur LogP prédite pour cette molécule [175]. Ils classifient chaque atome d'une molécule en fonction de ses atomes liés et de ses atomes voisins. Ils utilisent ensuite la formule

$$LogP(m) = \sum_i n_i a_i \tag{2.2}$$

pour prédire LogP. n_i est le nombre d'atomes de type i dans la molécule m et a_i la contribution à LogP pour un atome de type i.

Leur classification finale aboutit à 68 groupes d'atomes et est conçue de telle sorte que chaque atome d'une molécule soit assigné à une seule classe. Pour garantir la portabilité et la reproductibilité de leur méthode, les auteurs ont donné les codes SMARTS [174] correspondants à leur classification.

L'intérêt de leur méthode est que non seulement il s'agit d'un descripteur moléculaire qui prédit le LogP d'une molécule, mais ce faisant il assigne une contribution au LogP total pour chaque atome. Ceci permet d'obtenir une valeur atome centrée relative à l'espace hydrophobe pour chaque atome d'une molécule. Cette particularité est utile au logiciel ACPC.

Méthode	Discrétisation	AUC moy./méd.
Moro 2005	histogramme normalisé	
électrostatique[108]	d ∈ [1,13]Å, dx = 1Å	0.49 / 0.50
Broto 1984 stérique[18]	histogramme, dx = 0.2Å	0.58 / 0.57
Broto 1984 hydrophobe[18]	histogramme, dx = 0.2Å	0.64 / 0.63
ACPC-1.0 électrostatique[15]	linear-binning, dx=0.005Å	0.69 / 0.67

TABLE 2.1 – Comparaison des aires sous la courbe ROC (moyenne et médiane) de différentes méthodes de discrétisation du vecteur d'autocorrélation. dx est le pas de discrétisation. Expérience : 51 cibles choisies au hasard dans la DUD-E[113]. Trois ligands requêtes choisis au hasard pour chaque cible et score de $\text{Tversky}_{\text{ref}}$.

2.4.2 La fonction d'autocorrélation

Cette fonction est au cœur du logiciel ACPC. Si l'on s'en tient strictement à la définition de Todeschini et Consonni, appliquer la fonction d'autocorrélation à une molécule donne lieu à plusieurs descripteurs moléculaires (un pour chaque distance inter-atomique présente dans la molécule).

La fonction d'autocorrélation a été utilisée dans de nombreuses méthodes de chémoinformatique pour encoder une molécule représentée 2D ou 3D de façon rotation et translation invariante. Cette transformation permet d'éviter la phase complexe et coûteuse en temps de calcul de superposition des molécules nécessaire avant leur comparaison.

En 1980, Gilles Moreau et Pierre Broto ont proposé pour la première fois[107] d'utiliser la fonction d'autocorrélation sur le graphe d'une molécule afin d'encoder n'importe quelle valeur centrée sur les atomes. Ils ont ensuite utilisé leur descripteur dans des études de relation entre la structure et l'activité d'une molécule[18,19].

Parmi les nombreuses méthodes basées sur la fonction d'autocorrélation, on peut citer Atom-Type AutoCorrelation (ATAC) de Todeschini et co-auteurs[157]. Les paires d'atomes[22] et la méthode Chemically Advanced Template Search (CATS)[132]. CATS est disponible en 2D[132] et en 3D[40]. CATS travaille avec des types d'atomes généralisés, ce qui est assez proche de points pharmacophoriques : donneur ou accepteur d'hydrogène, chargé positivement ou négativement, hydrophobe. L'autocorrélation des charges partielles d'une molécule tridimensionnelle à été étudiée par le passé mais malheureusement via un

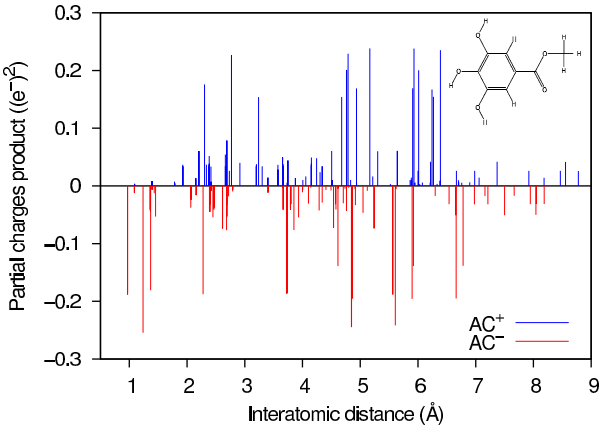


FIGURE 2.6 – Autocorrélogramme des charges partielles de la conformation 3D de plus basse énergie d'une molécule. La molécule est montrée en haut à droite en 2D.

encodage par histogramme[10]. On peut voir (tableau 2.1) l'effet d'un encodage par histogramme de la fonction d'autocorrélation. L'autocorrélation de propriétés à la surface d'une molécule à été étudiée dans des études utilisant de la régression numérique[108], de l'analyse des composantes principales[169], des cartes de Kohonen[169] et des réseaux de neurones[6] afin de générer des modèles de QSAR et faire de la prédiction d'activité biologique.

Nous donnons ici la formule de la fonction d'autocorrélation, telle qu'utilisée en chémoinformatique.

Soit M une molécule avec N atomes.

Soit $i = (x_i, y_i, z_i, q_i)$ l'atome à la position i $(1 <= i <= N)$ dans M avec les coordonnées (x_i, y_i, z_i) et la charge partielle q_i.

Soit d_{ij} la distance euclidienne entre les atomes i et j dans M.

Soit k une distance inter-atomique et $\delta_{kd_{ij}}$ le delta de Kronecker égale à un quand $k = d_{ij}$ et zéro partout ailleurs.

L'autocorrélation de la molécule M à la distance k peut s'écrire :

$$AC(M,k) = \sum_{i=1}^{N} \sum_{j=1}^{N} q_i q_j \delta_{kd_{ij}} \tag{2.3}$$

Dans la pratique, nous ignorons les valeurs de la fonction d'autocorrélation pour $k = 0$,

car nous les trouvons peu discriminantes. En effet, toutes les molécules ont un pic en $k = 0$ dans leur autocorrélogramme et il est proportionnel aux nombres d'atomes partiellement chargés de la molécule. Un autocorrélogramme est la représentation graphique de la fonction d'autocorrélation (figure 2.6).

Aussi, nous ne considérons qu'une seule fois une paire d'atomes (i, j), parce que considérer aussi la paire (j, i) ne fait que dupliquer de l'information déjà présente et ajoute un calcul inutile.

Le logiciel ACPC utilise donc la formule

$$ACPC(M,k) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} q_i q_j \delta_{k d_{ij}} \tag{2.4}$$

Comme nous souhaitons encoder la molécule M de la façon la plus fidèle possible, nous ne considérons pas un sous ensemble arbitraire des valeurs de k mais toutes les valeurs possibles de k pour la molécule M. Si M a N atomes, nous obtenons donc un ensemble de N(N-1)/2 contributions pour encoder M. Il est important de remarquer que cet encodage est non réversible. Étant donné un autocorrélogramme, il peut exister plusieurs molécules ayant le même. Par exemple, deux molécules qui sont l'image l'une de l'autre dans un miroir (des stéréoisomères) auront le même autocorrélogramme. Mais il s'agit à notre avis du seul prix à payer avec la fonction d'autocorrélation, afin de gagner l'avantage d'être rotation et translation invariant.

Les charges partielles sont un descripteur moléculaire particulièrement intéressant dans l'espace électrostatique. En effet, elles permettent le calcul du champ électrostatique que nous introduisons tout de suite.

2.5 Le champ électrostatique

Nous introduisons ici comment calculer le champ électrostatique autour d'une molécule (dans le vide, par soucis de simplicité). Ceci afin de donner une intuition de la complexité et du coût du calcul et ainsi justifier pourquoi nous avons parfois évité d'avoir à faire explicitement ce calcul dans nos travaux.

Si l'on a N charges électriques ponctuelles fixes q_i disposées dans l'espace (3D) en des points r_i, le champ électrostatique observé au point d'observation r est donné par la formule :

$$\vec{E}(r) = \frac{1}{4\pi\epsilon_0} \sum_{i=1}^{N} \frac{q_i}{\left\|\vec{R_i}\right\|^3} \vec{R_i} \tag{2.5}$$

où ϵ_0 est la permittivité diélectrique du vide et $\vec{R_i} = r - r_i$. La constante $\frac{1}{4\pi\epsilon_0}$ est aussi appelée constante de Coulomb et se note k_e.

On voit donc que le champ électrostatique autour d'une molécule est fonction des charges partielles de cette molécule, de leur distribution dans l'espace ainsi que de la permittivité diélectrique du milieu qui entoure cette molécule. De plus, si l'on veut comparer le champ électrostatique autour de molécules (ce qui est fait par le logiciel EleKit qui sera présenté plus tard), il faut au préalable superposer ces molécules. Il existe des méthodes éprouvées[56] et des implémentations libres[114,126,149] mais la tâche reste non triviale et a un coût. Pour des raisons de performance, il n'est pas nécessaire de comparer les champs électrostatiques de deux molécules en tous points de l'espace. On peut se contenter de comparer la distribution spatiale des charges qui génèrent ces champs. C'est ce que fait le logiciel ACPC qui sera présenté par la suite.

Le calcul explicite du champ électrostatique autour d'une molécule en présence d'un solvant est une tâche complexe et lourde en temps de calcul. En plus de la position des atomes et de leurs charges partielles, il nécessite de connaître la permittivité diélectrique du solvant et du corps dissous, les rayons des atomes[37,162] ainsi que la concentration ionique du solvant. Il existe des logiciels spécialisés tels qu'APBS[8] et DelPhi[115], développés par des biophysiciens, qui sont capables d'effectuer ce calcul de manière relativement efficace. Ces logiciels sont parallèles et peuvent traiter des macromolécules telles que le ribosome. Ils résolvent numériquement l'équation de Poisson-Boltzman[32,67]. Des développements récents[181] résolvent cette équation de manière semi analytique, ce qui permet d'aller plus vite et d'être plus précis numériquement.

Nous venons de voir que pour calculer un champ électrostatique, il faut connaître la valeur des charges partielles. Introduisons donc maintenant une méthode qui permet de

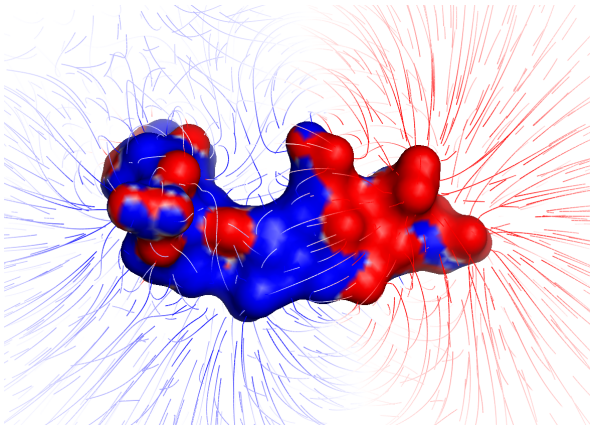


FIGURE 2.7 – Lignes du champ électrostatique autour de la protéine ligante intégrine (PDB:2VDO). La surface de la protéine et les lignes de champ sont colorées en fonction de la valeur du champ électrostatique. Le rouge est pour les valeurs négatives et le bleu pour les valeurs positives.

calculer ces charges.

2.5.1 Les charges partielles et le modèle de Gasteiger-Marsili

Nous mentionnons ici un modèle populaire et efficace pour le calcul des charges partielles assignées à chaque atome d'une molécule organique. Ce modèle donne de bons résultats avec le logiciel ACPC et est l'un des nombreux modèles (EEM[109], PEOE[49], MMFF94[61], QEq[120], QTPIE[23]) disponibles dans Open Babel[116].

L'apparition d'une charge partielle est due à la dissymétrie dans la distribution des électrons autour d'un atome, provoquée par les liaisons chimiques de cet atome lorsqu'il est au sein d'une molécule.

La distribution des électrons d'une molécule peut être calculée avec de la mécanique quantique[112]. Mais le coût de tels calculs ont poussé Johann Gasteiger et Mario Marsili à développer leur propre méthode empirique de calcul[49]. Leur méthode, qu'ils nomment « iterative Partial Equalization of Orbital Electronegativity » (abrégée en PEOE), utilise seulement le type des atomes ainsi que leur connectivité pour calculer les charges partielles d'une molécule. Typiquement, après seulement six itérations, leur méthode de calcul converge vers un résultat suffisamment précis.

Pour quantifier la similarité, un descripteur moléculaire (un vecteur) doit être combiné à une fonction de score ou une distance. Nous introduisons ci-après quelques scores et distances typiques.

2.6 Quelques mesures de similarité

Dans une section précédente, nous avons vu que la fonction d'autocorrélation est une façon d'encoder une molécule. Si l'on choisit ensuite un pas de discrétisation, il devient alors possible de stocker cet encodage dans un vecteur. Deux vecteurs peuvent être comparés via une fonction de score ou une distance. Nous listons ici quelques fonctions de score largement utilisées en chemoinformatique[40,149]. Pour des scores de corrélation (du moins robuste et plus simple à calculer au plus robuste mais plus compliqué) ainsi que leurs conditions d'utilisation, tels que Pearson, Spearman et le Tau de Kendall on se rapportera au livre « numerical recipes »[117].

Dans les formules suivantes A et B sont deux molécules encodées sous forme de vecteurs. Ces vecteurs ont n éléments et $A[i]$ dénote l'élément d'indice i ($1 <= i <= n$) dans le vecteur qui encode A.

2.6.1 Les distances d'Euclide et de Manhattan

Certains auteurs[40] utilisent la distance Euclidienne ainsi que la distance de Manhattan dont voici les formules :

$$d_{\mathrm{Euc}}(A,B) = \sqrt{\sum_{i=1}^{i=n}(A[i]-B[i])} \tag{2.6}$$

$$d_{\mathrm{Man}}(A,B) = \sum_{i=1}^{i=n}|A[i]-B[i]| \tag{2.7}$$

2.6.2 L'indice de Jaccard/Tanimoto

Dans la littérature chimique, on trouve très souvent mention du score de Tanimoto[151]. Le score est souvent utilisé avec une valeur seuil, dont le choix semble arbitraire. Ceci dit, plus général et antérieur au score de Tanimoto est l'indice de Jaccard[73] qui date de 1901 et qui peut s'énoncer pour deux ensembles alors que le score de Tanimoto n'est formulé

que pour des pairs de vecteurs de booléens. Nous présentons le score de Jaccard en premier car l'indice de Tversky présenté ensuite n'en est qu'une version paramétrée.

$$i_{\text{Jac}}(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.8}$$

Sous forme vectorielle, il s'énonce [40] :

$$i_{\text{Jac}}(A,B) = \frac{\sum_{i=1}^{i=n} A[i]B[i]}{\sum_{i=1}^{i=n}(A[i] + B[i] - A[i]B[i])} \tag{2.9}$$

Il est intéressant de noter que l'indice de Jaccard peut être transformé en une distance via l'application de la formule :

$$d_{\text{Jac}}(A,B) = 1 - i_{\text{Jac}}(A,B) \tag{2.10}$$

2.6.3 L'indice de Tversky

L'indice de Tversky est parfois appelé score d'inclusion. Il est dissymétrique, au contraire de l'indice de Jaccard/Tanimoto et de la distance euclidienne. Via l'insertion des paramètres α et β dans la formule de l'indice de Jaccard, on aboutit à l'indice de Tversky [164]. Ces paramètres permettent d'affecter un poids à une molécule et un poids moindre à l'autre molécule, en respectant les contraintes ($\alpha >= 0 \wedge \beta >= 0 \wedge \alpha + \beta = 1.0$). De manière générale, l'indice de Tversky s'énonce :

$$i_{\text{Tve}}(A,B) = \frac{|A \cap B|}{\alpha|A| + \beta|B| - |A \cap B|} \tag{2.11}$$

. Ce qui donne sous forme vectorielle :

$$i_{\text{Tve}}(A,B) = \frac{\sum_{i=1}^{i=n} A[i]B[i]}{\sum_{i=1}^{i=n}(\alpha A[i] + \beta B[i] - A[i]B[i])} \tag{2.12}$$

Deux formes spéciales de l'indice de Tversky sont particulièrement intéressantes. Si A est notre molécule requête et B une molécule candidate (issue d'une chimiothèque) on parle de $\text{Tversky}_{\text{ref}}$ si $\alpha = 1$ et de $\text{Tversky}_{\text{db}}$ si $\beta = 1$.

$$i_{\text{Tve}_{\text{ref}}}(A,B) = \frac{\sum_{i=1}^{i=n} A[i]B[i]}{\sum_{i=1}^{i=n}(A[i] - A[i]B[i])} \tag{2.13}$$

$\text{Tversky}_{\text{ref}}$ permet de trouver des molécules qui sont un sur-ensemble de la molécule requête A. Nous avons remarqué, ainsi que d'autres auteurs [69], que l'indice $\text{Tversky}_{\text{ref}}$ (ou un

Tversky avec α proche de un) est extrêmement puissant et donc recommandé pour les tâches de LBVS.

$$i_{\mathrm{Tve_{db}}}(A,B) = \frac{\sum_{i=1}^{i=n} A[i]B[i]}{\sum_{i=1}^{i=n}(B[i] - A[i]B[i])} \tag{2.14}$$

$\mathrm{Tversky_{db}}$ permet quant-à-lui de trouver des molécules qui sont un sous-ensemble de la molécule requête A.

Muni d'un vecteur décrivant une petite molécule (ou la géométrie d'une protéine) ainsi que d'une distance, on peut calculer des groupes d'éléments.

2.7 Le regroupement

Nous introduisons ici quelques notions sur le regroupement afin que le lecteur ait une idée de l'abondance et de la variété des techniques qui existent dans ce domaine. Le sujet étant vaste et pour ne pas nous éparpiller, nous mentionnons principalement les algorithmes combinatoires. Ces algorithmes travaillent directement sur les données observées et ne nécessitent pas la connaissance préalable d'une éventuelle loi de distribution qui serait suivie par les données.

Si l'on est en présence de C classes et de N éléments que l'on veut classifier, partitionner les éléments consiste à assigner chaque élément à une et une seule classe. Le logiciel Durandal, qui sera présenté par la suite, implémente de façon très efficace un algorithme de regroupement. Le goulet d'étranglement de nombreuses méthodes de regroupement est le calcul de la matrice de dissimilarité entre les modèles. En effet, pour N éléments on doit calculer $\frac{N(N-1)}{2}$ distances. Pour des conformations de protéines, on utilise classiquement la distance RMSD, qui donne la distance entre deux conformations de protéines après leur superposition optimale.

Soit p et q deux conformations de la même protéine superposées de façon optimale et comportant n atomes chacune. Soit p_i (resp. q_i les coordonnées du ième atome de p (resp. q). La formule du RMSD s'énonce :

$$\mathrm{RMSD}(p,q) = \sqrt{(\frac{\sum_{i=1}^{n}(p_i - q_i)^2}{n})} \tag{2.15}$$

De nombreuses méthodes existent pour calculer le RMSD [35,68,79,82,154], avec ou sans dé-

termination de la transformation rigide qui permet de superposer les structures de manière optimale. La plus performante à ce jour semble être la méthode de Douglas Theobald [154] dénommée QCP pour « Quaternion-based Characteristic Polynomial ». L'article de Theobald va jusqu'à compter le nombre maximum d'opérations en virgule flottante (FLOPS) nécessaires par différentes méthodes avant de conclure que QCP est la méthode la plus efficace. La méthode QCP est aussi numériquement stable, ce qui n'est pas le cas de toutes les méthodes.

En anglais, on dit « clustering » pour parler de regroupement mais le terme anglais est très souvent utilisé même par les francophones. Le clustering est un ensemble de techniques utilisées en analyse de données et en apprentissage automatique. Le but du clustering est de discerner des groupes dans des observations (on parle alors d'apprentissage non supervisé) ou d'assigner des observations à des groupes connus (on parle alors d'apprentissage supervisé). Il existe aussi de nouvelles techniques qui sont dites semi supervisées [27] mais nous n'entrerons pas dans ces développements récents.

Il faut remarquer que certains algorithmes forcent la formation de groupes, même si les données ne sont pas structurées en groupes. Certains algorithmes forcent aussi la création d'une hiérarchie, même si les données ne présentent pas une telle structure. Le choix de la distance, ainsi que de l'algorithme à utiliser est donc important et à choisir en fonction de la connaissance des données, du domaine ainsi que du problème à traiter. Pour une bonne référence anglaise gratuite en statistiques et en apprentissage artificiel, dont du clustering, on pourra se référer au livre « The Elements of Statistical Learning » [64]. Une bonne référence en français sur l'apprentissage automatique est le livre « Apprentissage artificiel » [27].

2.7.1 Les K-moyennes

En anglais l'algorithme s'appelle k-means [104]. On doit lui fournir en entrée le nombre K de clusters dans lesquels répartir les données.

Algorithm 1 Algorithme des K-moyennes.

0) assigner aléatoirement chaque observation à l'un des K clusters
1) calculer le représentant moyen de chaque cluster
2) assigner chaque élément au cluster dont il est le plus proche du représentant moyen
3) répéter les étapes 1) et 2) jusqu'à ce qu'il n'y ait plus d'élément qui change de cluster lors de l'étape deux.

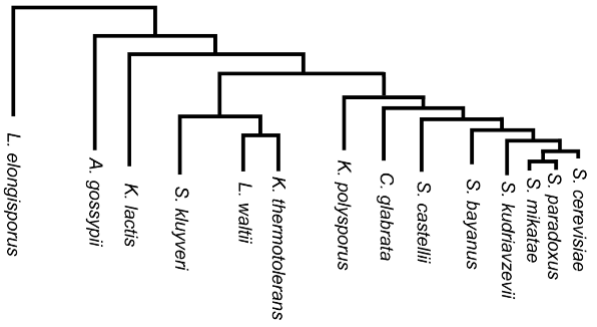


FIGURE 2.8 – Dendrogramme résultant de la classification de levures[168]. Image sous licence creative commons.

2.7.2 Les K-médoïdes

En anglais l'algorithme s'appelle k-medoids[80]. On doit lui fournir en entrée le nombre K de clusters dans lesquels répartir les données.

Médoïde : point le plus proche de la moyenne d'un ensemble de points.

L'algorithme des K-médoïdes est un algorithme plus robuste aux données aberrantes que l'algorithme précédant des K-moyennes. Malheureusement, cette robustesse se paie au prix d'un calcul plus lourd.

Algorithm 2 Algorithme des K médoïdes.

0) choisir aléatoirement K points distincts comme médoïdes (représentants de chacun des clusters)
1) assigner chaque élément au cluster du médoïde dont il est le plus proche
2) recalculer les médoïdes
3) répéter les étapes 1 et 2 jusqu'à ce qu'il n'y ait plus d'élément qui change de cluster lors de l'étape 1).

2.7.3 Le clustering agglomératif hiérarchique

L'algorithme commence avec chaque élément dans un cluster distinct (singleton). À chaque étape, les deux clusters les plus similaires sont unis en un seul. Ce jusqu'à obtenir un cluster unique regroupant tous les éléments.

Cet algorithme nécessite donc une mesure de dissimilarité entre deux clusters. En fonction de la mesure choisie, l'algorithme change de nom. Si la distance inter clusters est la distance moyenne entre les clusters, l'algorithme est nommé « group average (GA) agglomerative clustering ». Le « single linkage (SL) agglomerative clustering » utilise la plus petite distance entre deux membres de deux clusters distincts comme distance entre ces deux clusters [145]. SL a tendance à créer des clusters de large diamètre. Le « complete linkage (CL) agglomerative clustering » utilise quant-à-lui la plus grande distance entre deux membres de deux clusters distincts comme distance entre ces deux clusters [78]. CL a tendance à créer des clusters de petit diamètre. Le comportement de GA est un compromis entre celui de SL et de CL. Les clusters obtenus avec GA sont relativement compacts et éloignés les uns des autres. SL et CL sont bien moins coûteux à calculer comparé à GA. Le résultat de tels algorithmes peut être visualisé au moyen d'un diagramme appelé dendrogramme, qui montre quels sont les clusters à chaque niveau de la hiérarchie de regroupement (figure 2.8). L'algorithme de Ward [172] est une autre méthode populaire de regroupement agglomératif qui vise a minimiser la variance intra cluster.

Le pendant du clustering agglomératif est le clustering divisif. Au lieu de partir de N singletons pour arriver à un seul groupe, ces méthodes partent de un seul groupe pour finir avec N singletons.

2.7.4 Les cartes de Kohonen

Teuvo Kohonen à créé un réseau de neurones [84] qui permet de projeter un espace d'entrée de dimension N dans une carte de dimension deux (habituellement) qui préserve la topologie de l'espace d'entrée. Deux vecteurs proches dans l'espace d'entrée seront proches sur la carte de Kohonen. Une carte de Kohonen doit être entraînée (phase d'apprentissage) avant de pouvoir être utilisée pour faire de la classification (phase d'exploitation). On

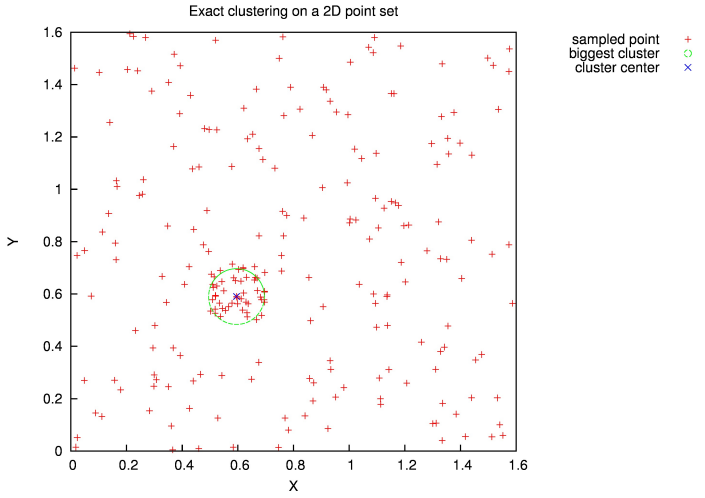


FIGURE 2.9 – Exemple de regroupement exact sur un ensemble de points. Les croix rouges sont les éléments à partitionner. La plus grosse partition trouvée est dans le cercle vert.

peut remarquer que Johann Gasteiger ainsi que ses collègues ont été des pionniers dans l'application de techniques d'intelligence artificielle, dont les cartes de Kohonen, à des problèmes de chimie[50,52,163,188].

2.7.5 Le regroupement exact

Présentons maintenant l'algorithme implémenté par le logiciel Durandal, qui sera présenté en section 3.2.

Algorithm 3 Algorithme de regroupement exact avec la distance seuil 'd'.

$e \leftarrow \{elements\}$
$m \leftarrow matrice\_dissim(e)$
while $e \neq \emptyset$ **do**
  $pgc \leftarrow plus\_grand\_cluster(e, m, d)$
  $sauvegarder(pgc)$
  $e \leftarrow e \setminus pgc$
end while

En plus de la grande diversité des algorithmes de regroupement, on trouve aussi de nombreuses variantes approximatives d'algorithmes connus. Souvent, les versions approximatives sont créées afin d'accélérer le calcul. Le logiciel Durandal implémente une version accélérée mais non approximée de l'algorithme dit de regroupement exact. Cet algorithme

(algorithme 3) ainsi qu'un exemple sur des données générées artificiellement est montré dans la figure 2.9. Cet algorithme ne nécessite pas un nombre de groupes à obtenir en entrée, mais nécessite une distance seuil qui sert à déterminer si deux éléments sont suffisamment proches pour être assignés au même groupe.

Même des algorithmes dits « non supervisés » peuvent avoir besoin d'une consigne en entrée, par exemple un nombre de clusters à obtenir ou une distance seuil. Le scientifique qui voudrait du tout automatique pourrait rester sur sa faim. Heureusement, il existe des méthodes basées sur des modèles (tel Mclust[45] disponible dans R[46,118]) qui permettent de déterminer automatiquement le nombre de clusters ainsi que leur composition. Mais on sort alors du domaine des algorithmes combinatoires et ces méthodes passent mal à l'échelle.

2.8 Le criblage virtuel *in-silico*

Combinés à une distance, les descripteur moléculaires peuvent être utiles pour chercher des molécules similaires. Ils peuvent donc s'avérer utiles lors du criblage par ordinateur d'une chimiothèque virtuelle.

Le criblage virtuel *in-silico* est une tâche de classification (figure 2.10). Les éléments à classifier sont des molécules et l'on est en présence de deux classes : les molécules actives sur la protéine ciblée et les molécules inactives. En situation réelle, la classe de chaque molécule est inconnue à l'avance. Aussi, en fonction du jeu de données utilisé lors de la validation scientifique d'une méthode, il se peut que seules certaines molécules aient été testées expérimentalement et détectées comme actives. Les molécules inactives sont souvent des leurres qui ont été générés automatiquement à partir des molécules actives[71,113]. Ceci peut introduire des erreurs ; certains leurres se révéleraient probablement actifs s'ils étaient testés expérimentalement.

Lors de la recherche de nouvelles molécules thérapeutiques, le criblage virtuel peut être utilisé en début du processus afin de sélectionner un petit ensemble de molécules qui vont être testées expérimentalement. Typiquement (hors contexte industriel), sur les millions de molécules d'une chimiothèque, entre 50 et 200 molécules seulement seront sélectionnées

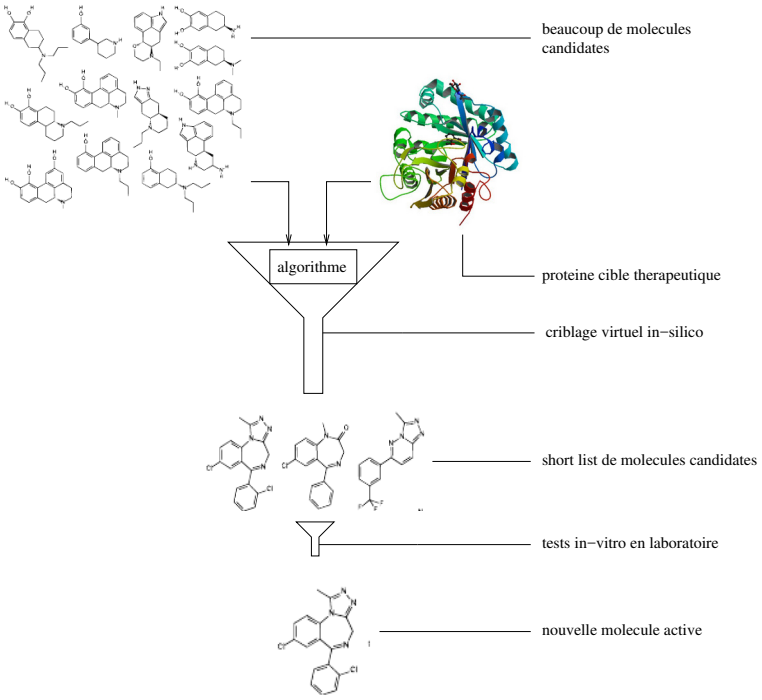


FIGURE 2.10 – Le criblage virtuel dans le contexte de la découverte de nouvelles molécules actives. Des méthodes informatiques permettent de prioriser certaines molécules afin qu'elles soient testées in-vitro en laboratoire.

après criblage virtuel, en fonction du budget alloué aux premiers tests en laboratoire.

2.8.1 L'aire sous la courbe ROC

La courbe ROC (« Receiver Operating Characteristic curve », figure 2.11) trace l'évolution du taux de vrais positifs (la prédiction dit vrai et c'est correct) en fonction du taux de faux positifs (la prédiction dit vrai mais c'est incorrect).

L'aire sous la courbe ROC (ASC) est une mesure comprise entre zéro et un de la qualité d'un classificateur. C'est une mesure importante lorsque l'on évalue une méthode de criblage virtuel. Citons ici quelques valeurs repères de l'ASC : un classificateur parfait aura une ASC de 1.0 et un classificateur aléatoire une ASC de 0.5. L'inverse du classificateur parfait aura une AUC de 0 (il n'assigne jamais un élément dans la classe adéquate). 1.0 et 0.5 sont deux repères importants : lors du développement d'un classificateur on doit essayer d'atteindre une ASC de 1.0 et si l'on développe une méthode dont l'ASC ne dépasse pas

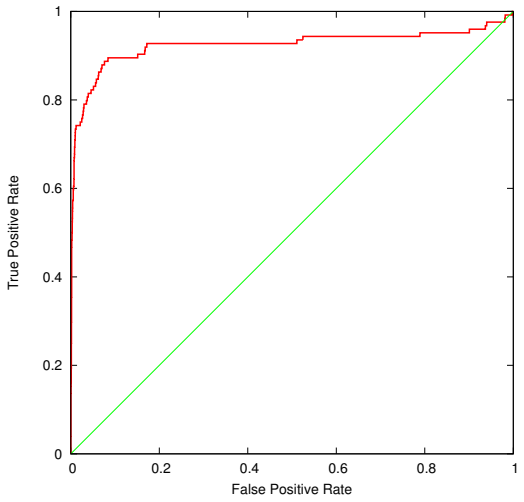


FIGURE 2.11 – Courbe ROC obtenue avec le logiciel ACPC-1.2 sur la protéine cible PDB:xiap. L'aire sous cette courbe ROC est de 0.92.

0.5, cela signifie que la méthode est aussi peu performante qu'un choix aléatoire.

2.8.2 Le facteur d'enrichissement

Une autre mesure intéressante en LBVS est le facteur d'enrichissement ($EF_{x\%}$). Il mesure l'amélioration du taux de molécules actives trouvées dans les premiers $x\%$ de molécules les mieux classées de la chimiothèque, par rapport au taux d'actives global (E_{tot}) de la chimiothèque. Par exemple, si un jeu de données contient globalement 10% de molécules actives mais qu'une méthode est capable de trouver 50% de molécules actives dans le top 5% des molécules ordonnées par le criblage ; la méthode considérée à un $EF_{5\%}$ de cinq. L'$EF_{x\%}$ est une mesure particulièrement intéressante en criblage virtuel puisque seule une petite fraction des molécules criblées les mieux classées seront testées expérimentalement. E_{tot} donne la probabilité de trouver une molécule active si l'on en choisit une au hasard.

$$E_{tot} = \frac{\|actives\|}{\|actives\| + \|inactives\|} \tag{2.16}$$

$$EF_{x\%} = \frac{E_{x\%}}{E_{tot}} \tag{2.17}$$

2.8.3 Quelques jeux de données pour le LBVS

Pour une revue sur les jeu de données utilisables pour l'évaluation à posteriori de méthodes de LBVS, on peut consulter Lagarde[89].

Sinon, pour ne citer que quelques jeux de données on pense a DUD[71], DUD-E[113] ainsi que la toute récente NRLiSt BDB[87,88]. DUD contient 2950 molécules actives ciblant 40 protéines distinctes. Pour chaque molécule active, 36 leurres avec des propriétés physico-chimiques similaires en terme de poids et de LogP calculé sont présentes mais ces molécules leurres ont une topologie différente de celle de la molécule active correspondante.

DUD-E est une version améliorée de DUD. DUD-E comprend 22886 molécules actives ciblant 102 protéines. Avec une moyenne de 224 ligands par protéine, DUD-E fournit aussi 50 molécules leurres par molécule active. Les molécules leurres ont là encore des propriétés physico-chimiques comparables aux molécules actives mais ont une topologie 2D différente.

La toute récente NRLiSt BDB contient 9905 molécules actives concernant 339 protéines qui sont toutes des récepteurs nucléaires. Le gros avantage de ce jeu de données sur ceux mentionnés précédemment est qu'ici on n'est pas obligé d'utiliser des leurres générés par ordinateur. En effet, les molécules actives sont classées en deux groupes : les molécules agonistes et les molécules antagonistes. Ce qui veut dire qu'on peut utiliser ce jeu de données en prenant les molécules agonistes comme molécules actives et les molécules antagonistes comme molécules leurres. Mais il s'agira cette fois ci bien de molécules qui ont toutes été testées expérimentalement et non de molécules supposées inactives comme c'est le cas avec DUD et DUD-E.

Tous les concepts qui ont été exposés précédemment nécessitent du temps de calcul. Il peut donc être utile de paralléliser et distribuer ces calculs.

2.9 Le calcul parallèle et distribué

Par la suite, nous allons présenter plusieurs logiciels. Certains de ces logiciels sont capables de s'exécuter en parallèle voire même de façon distribuée ou en combinant les deux modes pour aller encore plus vite.

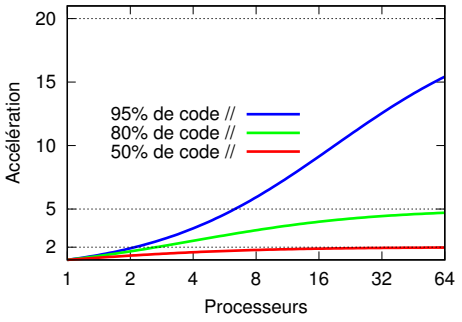


FIGURE 2.12 – La loi d'Amdahl dicte le facteur maximal d'accélération que l'on peut obtenir en fonction de la proportion de code qui s'exécute en parallèle et du nombre de processeurs utilisés. Les lignes horizontales pointillées sont les limites pour chacun des cas (50, 80 ou 95% du code s'exécutant en parallèle) si on considère une infinité de processeurs.

Exécution parallèle : le logiciel s'exécute sur un seul ordinateur mais utilise plusieurs processeurs de cet ordinateur en même temps. L'accélération ainsi obtenue obéit à la loi d'Amdahl (figure 2.12 et formule 2.9).

Certains logiciels multithreads, ainsi que la plupart des logiciels multi processus s'exécutent en parallèle. Par contre, certains logiciels multithreads ne donnent que l'illusion du parallélisme. Ils n'utilisent en fait qu'un seul processeur lors de leur exécution, c'est le cas de la plupart des interfaces graphiques utilisateur. Le but est alors que l'interface soit plus réactive vis à vis de l'utilisateur.

De nos jours, la plupart des ordinateurs et même les téléphones portables ont au moins deux processeurs. Une bonne station de travail peut en avoir de 16 à 32, à des prix abordables pour le monde professionnel. Ces ordinateurs sont la plupart du temps connectés via un réseau. Il est donc normal que des logiciels modernes soient conçus pour utiliser au mieux ces processeurs parallèles et cette infrastructure de communication.

La loi d'Amdahl[3] :

$$a = \frac{1}{1 - p + \frac{p}{s}} \tag{2.18}$$

a est le facteur d'accélération maximal atteignable sur une machine parallèle en fonction du nombre de cœurs (s) exécutant la section parallèle et de la fraction (p) de code dans la

section parallèle.

Par implication de la loi d'Amdahl, un programme qui passe bien à l'échelle dans le cas parallèle doit avoir une proportion maximale de code qui s'exécute en parallèle. La limite idéale étant que le programme n'a pas de portion de code qui s'exécute de manière séquentielle. C'est malheureusement impossible dans la pratique, étant donné qu'il y a toujours une portion de code, aussi petite soit elle, qui est nécessaire pour initialiser le programme avant de rentrer dans la portion de code parallèle. Dans certains cas, il y a aussi une section de code séquentiel en fin de programme, afin de finaliser le résultat du calcul. Certains programmes nécessitent aussi des communications entre processus en cours de calcul, ou l'accès à une ressource partagée (telle que le disque dur ou le noyau du système d'exploitation), ce qui fait perdre en parallélisme.

Exécution distribuée : le logiciel s'exécute en même temps sur plusieurs ordinateurs connectés via un réseau.

Pour que la parallélisation soit bonne dans le cas d'un programme distribué, il faut que les messages à échanger soient les moins nombreux possibles et de taille minimale. Ceci afin de minimiser la latence des communications. Le cas idéal est donc d'avoir un calcul distribué qui n'envoie pas de messages en cours de calcul. Les tâches sont alors indépendantes. Le logiciel PAR, qui sera présenté par la suite, permet d'exécuter de façon parallèle et distribuée des tâches indépendantes. PAR limite le nombre de messages. Par exemple, lorsqu'un « esclave » envoie un résultat au « maître » du calcul, ce message sert aussi de requête pour une nouvelle tâche de calcul.

Lorsqu'on veut avoir accès à une grande puissance de calcul, un bon moyen est d'utiliser un supercalculateur. Sur une telle machine, il est possible de lancer des calculs parallèles et/ou distribués ; par exemple en utilisant la librairie MPI [44,53]. Un autre moyen est d'utiliser une « desktop grid » telle BOINC [4] ou XtremWeb [41]. Mais, paradoxalement, utiliser un supercalculateur est parfois lent et incommode. Il faut préparer son expérience, la soumettre, attendre qu'elle démarre puis attendre qu'elle termine. Sur certains systèmes, on doit même attendre après la fin de l'expérience que les données soient rapatriées sur un ordinateur hors du supercalculateur (appelé une frontale). Sur un supercalculateur trop utilisé, le temps d'attente est fonction du nombre de processeurs que l'on demande. Une petite

expérience, utilisant peu de processeurs pendant un temps court, démarre vite. Par contre, dès lors que l'on demande une portion significative du nombre maximal de processeurs, le temps d'attente peut être de plusieurs heures voir de plusieurs jours. Ce mode d'interaction avec un supercalculateur est problématique. D'une part, préparer une expérience peut être compliqué, par exemple à cause de l'absence d'un système de fichier distribué. Il faut alors créer des scripts spécialisés pour interagir avec le supercalculateur, ne serait-ce que pour le transfert des données. Mais surtout, ces temps d'attente font que la boucle d'interaction de l'expérimentateur avec le résultat de ses expériences est trop lente. La moindre erreur peut coûter des heures d'attente. Une catégorie d'utilisateurs particulièrement impactée par ce type d'interaction est celle des chercheurs en dynamique moléculaire. Leurs expériences demandent beaucoup de temps de calcul et peuvent accaparer de nombreux processeurs.

Avant de passer aux résultats publiés, récapitulons d'abord dans la section qui suit les objectifs visés par cette thèse.

2.10 Objectifs

Calcul parallèle et distribué : nous souhaitons accélérer la boucle d'interaction d'un expérimentateur avec ses expériences computationnelles. Un logiciel qui permettrait de démarrer une expérience sans attendre et en utilisant tous les ordinateurs sur lesquels l'utilisateur à un accès distant serait utile à de nombreux scientifiques [2]. Cet objectif est traité dans la section 2.1 avec le logiciel PAR.

Regroupement rapide : afin d'aider nos collègues qui font de la prédiction de structures de protéines et analysent des trajectoires de dynamique moléculaire, nous souhaitons accélérer le regroupement de modèles de protéines. Cet objectif est traité dans la section 3.2 avec le logiciel Durandal.

Comparaison dans l'espace électrostatique : nous souhaitons mesurer la similarité dans l'espace électrostatique entre une petite molécule et la protéine qu'elle est censée remplacer sur une interface protéine-protéine. Une telle mesure serait utile lors de la conception de molécules destinées à perturber une interaction protéine-protéine. Cet

2. Le logiciel GNU parallel [150] est apparu peu après nos travaux.

objectif est traité dans la section 3.3 avec le logiciel EleKit.

Criblage virtuel in-silico basé sur des ligands : nous souhaitons concevoir une méthode rapide pour comparer des molécules dans l'espace électrostatique [3]. Une méthode qui ne nécessite pas de superposer les molécules au préalable permettrait de faire cette comparaison plus vite. Cet objectif est traité dans la section 3.4 avec le logiciel ACPC.

3. Il existe des logiciels commerciaux à cet effet ; par exemple ROCS et EON [111] d'Openeye peuvent superposer puis comparer les champs électrostatiques de petites molécules.

Chapitre 3

Résultats publiés

Dans les pages suivantes, nous présentons différents logiciels innovants en bioinformatique structurale, chémoinformatique et calcul distribué. Chacun de ces logiciels a fait l'objet d'au moins une publication scientifique dans un journal revu par des experts. Certains de ces logiciels ont aussi fait l'objet d'un poster présenté lors d'une conférence en cas d'évolution majeure ou pour souligner un point particulièrement intéressant.

3.1 Exécution parallèle et distribuée de commandes indépendantes

Les bioinformaticiens s'attaquent à des problèmes de plus en plus gourmands en temps de calcul et en espace de stockage. Dans le même temps, les processeurs deviennent de plus en plus parallèles. De nos jours, une machine de bureau standard peut facilement avoir de huit à seize coeurs de calcul. Les tâches de calculs de type « paquet de tâches » sont souvent rencontrées par les chercheurs en sciences computationnelles (bioinformaticiens, chemoinformaticiens, etc.). Un paquet de tâches consiste en un ensemble de tâches indépendantes les une des autres et qui ne communiquent pas entre elles en cours de calcul.

Le logiciel que nous proposons, PAR, permet d'exécuter efficacement de façon parallèle et distribuée un paquet de tâches. PAR accepte même que de la puissance de calcul soit ajoutée en cours d'exécution. PAR est destiné aux processeurs multi-coeurs et aux clusters de calcul relativement petits (jusqu'à quelques dizaines de noeuds de calcul). PAR à été utilisé en production avec jusqu'à 128 coeurs et a servi à accélérer la validation scientifique de la plupart des logiciels présentés dans cette thèse.

Nous avons mesuré l'accélération obtenue avec PAR dans des cas d'utilisation réels en bioinformatique structurale. En mode hybride parallèle et distribué (utilisant 16 machines et 4 processeurs par machine) PAR permet de terminer une expérience computationnelle entre 45 et 52 fois plus vite que si l'on n'utilisait qu'un seul processeur.

BIOINFORMATICS APPLICATIONS NOTE Vol. 26 no. 22 2010, pages 2918–2919
doi:10.1093/bioinformatics/btq542

Structural bioinformatics Advance Access publication September 23, 2010

PAR: a PARallel and distributed job crusher

Francois Berenger[1,*], Camille Coti[2] and Kam Y. J. Zhang[1]

[1]Zhang Initiative Research Unit, Advanced Science Institute, RIKEN, 2-1 Hirosawa, Wako, Saitama 351-0198, Japan and [2]HPC Group, Department of Mathematics, Iowa State University, Ames, IA 50011-2251, USA

Associate Editor: Anna Tramontano

ABSTRACT

Summary: Bioinformaticians are tackling increasingly computation-intensive tasks. In the meantime, workstations are shifting towards multi-core architectures and even massively multi-core may be the norm soon. Bag-of-Tasks (BoT) applications are commonly encountered in bioinformatics. They consist of a large number of independent computation-intensive tasks. This note introduces PAR, a scalable, dynamic, parallel and distributed execution engine for Bag-of-Tasks. PAR is aimed at multi-core architectures and small clusters. Accelerations obtained thanks to PAR on two different applications are shown.

Availability: PAR is released under the GNU General Public License version three and can be freely downloaded (http://download.savannah.gnu.org/releases/par/par.tgz).

Contact: berenger@riken.jp

Received on July 20, 2010; revised on September 7, 2010; accepted on September 19, 2010

1 INTRODUCTION

Bioinformaticians are significant high-performance computing users, in particular for simulations of biologic phenomena. On the other hand, the available hardware is not only getting faster but also much more parallelized (Intel publicly reported working on 80 cores prototype chips in 2007). In this context, most bioinformaticians could benefit from an easy-to-use software to harness such computing power.

The focus of this note is Bag-of-Tasks (BoT) applications execution. As the name suggests, BoT applications can be seen as a bag, filled with tasks to do, each being independent from all the others. A middle-ware for BoT applications is called a *job crusher*. It has to consist of at least a server component connected to a set of clients.

This note introduces PAR, a parallel and distributed job crusher working in *pull* mode and inspired by desktop grid platforms. Workers *join* the computation and can be added dynamically at run-time; the server delivers tasks to workers available at a given moment. PAR is actually a transposition of some concepts and features from previous distributed middle-ware to small HPC clusters and multi-core workstations.

This article is organized as follows: Section 2 presents an overview of related projects and technologies used in bioinformatics. Section 3 presents two examples using PAR to illustrate scalability. The last section lists upcoming enhancements.

*To whom correspondence should be addressed.

2 RELATED PROJECTS

A wide variety of tools and technologies have been used over the last two decades in bioinformatics. While PAR is a user-level tool with its own niche, it has some limitations. At the cost of a little more complexity, some of the tools listed hereafter allow fair share of resources, stronger reliability and even faster job or data throughput.

At the programming level, the Message-Passing Interface [MPI, Forum (1994)], CORBA (Object Management Group, 1998) or even MapReduce (Dean and Ghemawat, 2004) are noteworthy technology candidates.

MPI has become the *de facto* standard for programming highly parallel applications. It has been used in computational genomics (Swain *et al.*, 2005) and in molecular dynamics (de Lomana *et al.*, 2008; Johnston *et al.*, 2005).

For applications following a client-server model, CORBA can be used. Handling of genome maps has successful examples (Hu *et al.*, 1998; Jungfer and Rodriguez-Tomé, 1998).

For data-intensive applications, MapReduce and its open source implementation Hadoop (http://hadoop.apache.org) are more appropriate. They unleash operations over huge amounts of data and were used recently in sequence alignment (Sadasivam and Baktavatchalam, 2010).

However, at the application level, Desktop Grids (DG) are closer to the focus of this note. A server distributes tasks to workers located on machines that do not communicate with each other, potentially anywhere on the Internet. Condor (Litzkow *et al.*, 1988), XtremWeb (Fedak *et al.*, 2001) and BOINC (Anderson, 2004) are three platforms for highly parallel, multi-user applications. One of the best-known DG project in bioinformatics is probably Folding@home (Beberg *et al.*, 2009).

Like Hadoop and unlike most DG, PAR is designed to be used exclusively on *private* resources. PAR's ideal scale is then smaller than what DG systems usually target, but this permits a lower latency. For simplicity, PAR uses pull-driven task distribution. This removes the need for a complex software component (called a scheduler) and also allows to scale smoothly even in large, dynamic and heterogeneous environments. In addition, PAR never requires administrator privileges and is only run on-demand.

3 EXAMPLE USE

The first example experiment consists of computing Alpha Carbons Root Mean-Square Deviation after optimal superposition, noted $C_\alpha RMSD_{opt}$ hereafter, on one thousand *ab initio* generated structures for the protein target 256B. Distances between proteins are computed using the software from (Zhang and Skolnick, 2004). The second experiment performs molecular replacement (MR), a

 © The Author 2010. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oxfordjournals.org

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

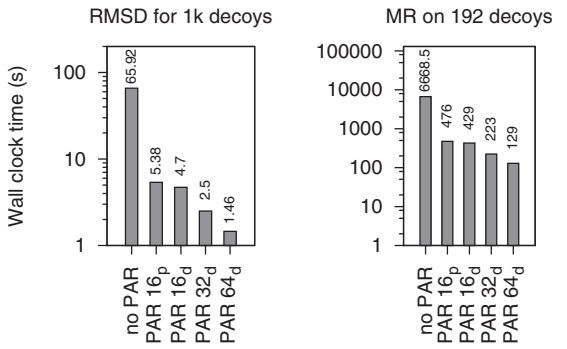


Fig. 1. Experiments accelerated by PAR (N_p: N parallel CPUs; N_d: N distributed CPUs).

method of solving the phase problem in X-ray crystallography using homologous structures, on a set of 192 decoys for the protein target 1m6t. We present the time elapsed with and without using PAR. PAR in parallel mode uses several cores of a given computer while the distributed mode uses distinct computers. The current implementation of PAR is known to work well with up to 16 and 64 CPUs in parallel and distributed mode, respectively.

Prior to timing experiments, needed programs and data were copied to each machine by the user. During experiments, PAR was started in server mode with a list of commands to execute. Workers were started soon after the server, but could have joined the computation later if we were not interested in the shortest completion time. The Unix 'time' command was used and averaged over two trials to measure the real time spent by PAR to complete all tasks. Unlike previous job crushers, PAR server's life cycle is only tied to the application's execution time (no Unix *daemon* involved) and PAR runs only in user space.

Results are shown in Figure 1. The first bar is the real time elapsed when not using PAR. The second bar is the time spent when using PAR in parallel mode, and the following bars are durations in distributed mode. On a CPU-intensive task and when using 16 CPUs, the speedup obtained by PAR can be as high as 14.01 in the parallel case and 15.54 in the distributed one. Lower performance of the parallel version is attributed to Python's problem with multi-thread applications (the Python interpreter uses a global lock mechanism shared by all threads). We can see that the application scales remarkably well. The overhead due to communications between workers and the master is very small, and this allows for an effective use of the parallel hardware with minimum effort required on the user's side.

4 FURTHER DEVELOPMENTS

PAR can be used on network of Unix-like workstations. It can take advantage of a Network shared File System (NFS). However, because of poor NFS performances, data-intensive tasks should be computed on top of a Distributed File System (DFS). As DFS are still rare even within clusters, we envisage to plug in such a functionality into PAR. A prototype has been implemented but is still in experimental stage.

PAR should integrate fault-tolerance policies, in order to be used safely even with more workers over longer periods, and with minimal overhead.

Furthermore, compression could be added to speedup communications. Encryption would be similarly easy to add and would allow PAR to be used over untrusted networks.

Finally, features can be added for large-scale experiments. For example, requesting groups of jobs instead of one at a time would lower the load on the server part. Allowing PAR to run both as a server and as a client would allow it to be deployed in layers, which could be used to connect several clusters together and increase scalability. Requests and contributions from users are also considered.

ACKNOWLEDGEMENTS

We thank all the PAR users, especially early ones like Rojan Shrestha for providing feedback and useful feature requests. We wish to thank RIKEN, Japan, for an allocation of computing resources on the RIKEN Integrated Cluster of Clusters (RICC) system.

Funding: 'Initiative Research Unit' program from RIKEN, Japan.

Conflict of Interest: none declared.

REFERENCES

Anderson,D.P. (2004) BOINC: a system for public-resource computing and storage. IEEE Computer Society, Pittsburgh, PA, USA, pp. 4–10.

Beberg,A.L. *et al*. (2009) Folding@home: lessons from eight years of volunteer distributed computing.

Dean,J. and Ghemawat,S. (2004) Mapreduce: simplified data processing on large clusters. In *OSDI'04*, USENIX Association, Berkeley, CA, USA.

de Lomana,A.L.G. *et al*. (2008) Optimal experimental design in the modelling of pattern formation. *ICCS'08*, **5101**, 610–619.

Fedak,G. *et al*. (2001) Xtremweb: a generic global computing system. In *CCGRID'01*, IEEE Computer Society, pp. 582–587.

Forum,M.P.I. (1994) MPI: a message-passing interface standard. *Technical Report UT-CS-94-230*. Department of Computer Science, University of Tennessee, Knoxville, Tennessee.

Hu,J. *et al*. (1998) Design and implementation of a corba-based genome mapping system prototype. *Bioinformatics*, **14**, 112–120.

Johnston,M.A. *et al*. (2005) Framework-based design of a new all-purpose molecular simulation application: the adun simulator. *J. Comput. Chem.*, **26**, 1647–1659.

Jungfer,K. and Rodriguez-Tomé,P. (1998) Mapplet: a corba-based genome map viewer. *Bioinformatics*, **14**, 734–738.

Litzkow,M. *et al*. (1988) Condor - a hunter of idle workstations. In *ICDCS'88*. IEEE-CS Press, pp. 104–111.

Object Management Group (1998) *The Common Object Request Broker: Architecture and Specification*. Version 2.3. Object Management Group, Framingham, MA, USA.

Sadasivam,G.S. and Baktavatchalam,G. (2010) A novel approach to multiple sequence alignment using hadoop data grids. In *MDAC '10*. ACM, New York, NY, USA, pp. 1–7.

Swain,M. *et al*. (2005) Modeling gene-regulatory networks using evolutionary algorithms and distributed computing. *CCGRID'05*, **1**, 512–519.

Zhang,Y. and Skolnick,J. (2004) Scoring function for automated assessment of protein structure template quality. *Proteins Struct. Funct. Bioinformatics*, **57**, 702–710.

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

3.2 Regroupement rapide

Le regroupement (« clustering » en anglais) est utilisé afin de présenter une vue simplifiée d'une collection de données. Les items similaires sont regroupés afin de permettre à l'utilisateur de focaliser son analyse sur ces groupes et leurs représentants caractéristiques. Par exemple, il peut s'agir d'une large série de géométries d'une protéine où beaucoup de géométries sont pratiquement identiques, dans le sens où elles appartiennent au même bassin d'attraction énergétique définissant une « conformation » peuplée de la protéine. De même, on peut regrouper des protéines en familles, selon leur séquence, la structure de leurs sites actifs, etc.

Calculer la matrice des distances entre tous les items pour un jeu de données contenant de nombreux items est coûteux en temps de calcul. L'ordre de grandeur de la complexité d'une telle tâche est $O(n^2)$; n étant la taille du jeu de données analysé.

En général, seul un nombre réduit de géométries de protéines sont partitionnées après un filtrage initial via un seuil d'énergie choisi arbitrairement. L'énergie est donnée pour chaque modèle par le logiciel qui a généré les modèles, par exemple Rosetta [119] ou TASSER [178]. Les modèles de plus basse énergie sont supposés être ceux de meilleure qualité.

Un algorithme de regroupement rapide pour de grands jeux de données est bénéfique au domaine de la prédiction de structure de protéines et à l'analyse de résultats de simulations de dynamique moléculaire. Une trajectoire de dynamique moléculaire comporte elle aussi un grand nombre de modèles de protéines.

Avec le logiciel Durandal [1] , nous proposons une méthode utilisant la propagation de contraintes géométriques afin d'accélérer le regroupement, sans introduire d'approximation dans la mesure de distance entre protéines ni dans l'algorithme. Notre méthode peut être utilisée avec toute métrique. Les métriques coûteuses à calculer mais qui ont des bornes hautes et basses faciles à calculer bénéficient au maximum de cette méthode.

1. Durandal est un outil officiel de la suite logicielle pour la cristallographie Phenix [1] (`www.phenix-online.org`).

Nous avons comparé la précision de notre méthode face aux résultats du logiciel SPICKER[185] sur 40 grands jeux de données générés par le générateur de modèles de protéines I-TASSER[179]. Nous avons aussi fait des tests de performance sur six protéines cibles du jeu de données « semfold ». Dans nos tests, notre méthode permet de choisir une conformation plus proche de la protéine native dans 63% des cas comparé au logiciel SPICKER. Notre méthode a aussi montré qu'elle est significativement plus rapide qu'un autre logiciel de regroupement rapide appelé Calibur[98]. Dans certains cas, la vitesse de notre méthode surpasse même la vitesse d'une méthode approximative (SCUD[97]).

BIOINFORMATICS ORIGINAL PAPER

Vol. 27 no. 7 2011, pages 939–945
doi:10.1093/bioinformatics/btr072

Structural bioinformatics

Advance Access publication February 9, 2011

Entropy-accelerated exact clustering of protein decoys

Francois Berenger, Yong Zhou, Rojan Shrestha and Kam Y. J. Zhang*

Zhang Initiative Research Unit, Advanced Science Institute, RIKEN, 2-1 Hirosawa, Wako, Saitama 351-0198, Japan

Associate Editor: Anna Tramontano

ABSTRACT

Motivation: Clustering is commonly used to identify the best decoy among many generated in protein structure prediction when using energy alone is insufficient. Calculation of the pairwise distance matrix for a large decoy set is computationally expensive. Typically, only a reduced set of decoys using energy filtering is subjected to clustering analysis. A fast clustering method for a large decoy set would be beneficial to protein structure prediction and this still poses a challenge.

Results: We propose a method using propagation of geometric constraints to accelerate exact clustering, without compromising the distance measure. Our method can be used with any metric distance. Metrics that are expensive to compute and have known cheap lower and upper bounds will benefit most from the method. We compared our method's accuracy against published results from the SPICKER clustering software on 40 large decoy sets from the I-TASSER protein folding engine. We also performed some additional speed comparisons on six targets from the 'semfold' decoy set. In our tests, our method chose a better decoy than the energy criterion in 25 out of 40 cases versus 20 for SPICKER. Our method also was shown to be consistently faster than another fast software performing exact clustering named Calibur. In some cases, our approach can even outperform the speed of an approximate method.

Availability: Our C++ software is released under the GNU General Public License. It can be downloaded from http://www.riken.jp/zhangiru/software/durandal_released.tgz.

Contact: kamzhang@riken.jp

Received on November 30, 2010; revised on January 20, 2011; accepted on February 3, 2011

1 INTRODUCTION

Protein structure prediction from amino acid sequence generally involves the search for the lowest energy conformation. This is based on Anfinsen's hypothesis that the native state of a protein is at the global minimum in free energy (Anfinsen, 1973). Sometimes, the predicted structure with the lowest energy might not be the closest to the native structure due to imperfections in the free energy function used. It has been shown that clustering can be used to identify the best structure among many decoys (Shortle *et al.*, 1998; Zhang and Skolnick, 2004a). This is based on the hypothesis that there are a greater number of low-energy conformations surrounding the correct fold than there are surrounding low-energy incorrect folds. In order to fold efficiently and retain robustness to changes in amino acid sequence as well as tolerance to structural perturbations, proteins may have evolved a native structure situated within a broad basin of low-energy conformations (Shortle *et al.*, 1998).

The exact clustering algorithm can be described briefly as two repeating steps. First, the cluster containing the structure with the maximum number of neighbors within a predefined cutoff value is found. Second, this cluster is removed from the set remaining to be clustered. Subsequent clusters are found by iterating these steps until the remaining set is empty. Two of the most successful *ab initio* protein folding engines, Rosetta (Das and Baker, 2008) and I-TASSER (Wu *et al.*, 2007), use exact clustering in their protocol (Bonneau *et al.*, 2001; Raman *et al.*, 2009; Skolnick, 2006; Zhang and Skolnick, 2004a, b; Zhang *et al.*, 2005). In the context of protein folding, exact clustering is used to reduce the population of *in silico* generated models (also called 'decoys') at the end of a folding simulation. As there is an imperfect correlation between energy functions used during the folding process and distance to the native structure, clustering can choose structures that are closer to native better than selecting the lowest energy decoy (Shortle *et al.*, 1998). For a good introduction to clustering, the reader is referred to Jain *et al.* (1999) and Hastie *et al.* (2009).

The earliest publication concerning clustering in the context of protein folding decoy identification seems to be from Shortle *et al.* (1998). Clustering after energy filtering identified conformations close to the native structure better than when using energy as the sole selection criterion.

By using clustering and a variant of RMSD[1], less sensitive to protein length in the SCAR software, Betancourt and Skolnick could decide if a protein folding simulation need more sampling of the conformational space (Betancourt and Skolnick, 2001). If sampling was enough, they could create representative structures of the different fold families that were discovered.

In the ABLE folding engine, Ishida *et al.* used URMSD[2] and Kohonen's self-organizing maps (Kohonen, 1998) in the clustering step of their folding protocol (Ishida *et al.*, 2003). Some improvements over results obtained by Rosetta were reported.

To identify the best decoys from protein folding simulations, Zhang and Skolnick tested their SPICKER program on 1489 protein targets (Zhang and Skolnick, 2004a). Because of the limited computer memory, SPICKER first shrinks the decoy set to 13k members prior to clustering. A good correlation between the cluster density of the biggest cluster and RMSD to native of the cluster representative is shown. The construction of a representative decoy for a given cluster in SPICKER is quite unique: it is an average of all

*To whom correspondence should be addressed.

[1]In the rest of this article and unless otherwise mentioned, RMSD always means Root-Mean-Square Distance after optimal superposition. C_αRMSD stands for RMSD considering only C_α atoms.

[2]URMSD stands for unit-vector RMSD. It is a non protein-length biased version of the popular C_αRMSD (Kedem *et al.*, 1999).

© The Author 2011. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oup.com

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

the cluster members (called a 'centroid'). This procedure is reported to improve RMSD to native as opposed to simply choosing the cluster center among cluster members, albeit it can create a structure that needs some corrections prior to refinement. In 90% of the cases, if the highest cluster density (a measure of cluster compactness) was greater than 0.1, one of the top five cluster centroids had an RMSD to native below 6 Å. In SPICKER, the cluster cutoff value is refined automatically based on decoys distribution. Its value is fixed once the first cluster size has reached a given percentage of the whole decoy set or once a stop value is reached.

Faster clustering of decoys was investigated by Li and Zhou with SCUD (Li and Zhou, 2005). Computing a distance matrix takes considerable time [O(n^2) complexity], so Li and Zhou used an upper bound of RMSD called reference RMSD (rRMSD) to avoid computing the optimal superposition of many decoy pairs. A 9-fold increase in speed over brute force RMSD-based methods was reported. The obtained clusters still had a high similarity to the ones obtained when using RMSD. A strong correlation between RMSD and rRMSD was shown. SCUD uses an automatic threshold finding technique that increases the cluster cutoff until the three biggest clusters are statistically meaningful.

Most researchers working on decoy identification are using exact clustering but Gront and Kolinski used hierarchical clustering. With the Hierarchical Clustering of Protein Models software (HCPM; http://biocomp.chem.uw.edu.pl/HCPM), partitioning of the conformational space that was sampled during folding is possible. HCPM can act as a data reduction filter and create libraries of decoys with a variety of low-energy conformations (Gront and Kolinski, 2005; Gront *et al.*, 2005). HCPM was also used to identify structures with a native-like topology in the study of folding pathway by multiscale modeling (Kmiecik and Kolinski, 2008). HCPM incorporates a heuristic to automatically choose a clustering threshold to obtain clusters of reasonable sizes.

A model that can predict the number of decoys necessary to obtain a given low RMSD value from native was proposed by Li (2006). The model can also be trained on a few decoys to predict the minimum RMSD that would be present in a larger set of decoys. It can also reliably estimate the fraction of decoys in the largest cluster as a function of cutoff value.

Handling large amounts of decoys should allow us to find higher quality models, compared to doing data reduction prior to clustering (Zhang and Skolnick, 2004a). In Calibur (http://sourceforge.net/projects/calibur/), Li and Ng try to handle quickly large datasets (Li and Ng, 2010). Experiments with several tens of thousands of decoys are reported. As soon as the decoy set is larger than 4k decoys, Calibur is faster than SPICKER. The quality of Calibur results seems to be better or at least equal to that of SPICKER. Calibur's algorithm is a rather intricate assembly of three strategies. First, outlier decoys detected by a statistical test are filtered out. Second, cheap to compute lower and upper bounds of RMSD are used as much as possible. Third, the metric property of RMSD (Steipe, 2002) is used to avoid many distance computations. Calibur's default strategy for threshold finding is x-percentile based and deduced from statistics on the dataset.

We present here the fastest method (to the best of our knowledge) to implement exact clustering. Our method focus on efficient distance matrix initialization, the most time-consuming part of exact clustering. This initialization step is mandatory in order to enumerate clusters. We have implemented our method in a software called Durandal, in reference to a mighty sword from medieval French legends.

2 METHODS

2.1 Overview

Our approach to accelerate exact clustering is based on one basic idea. By taking advantage of the metric property of RMSD (Steipe, 2002), measuring all decoys versus a few references allows to have a coarse idea of how decoys are distributed in the conformational space. It allows to prune out superfluous computations by discovering which distances are not necessary to measure exactly. In the case of exact clustering, knowing only a range that the distance will fall into is enough in many cases. This range can be approximated and narrowed down using only addition or subtraction, which is several orders of magnitude faster than computing an RMSD.

In exact clustering (Algorithm 1), a cutoff value[3] is needed as an input parameter. The algorithm computes for each pair of decoys if it has an RMSD less-than-or-equal-to the cutoff value or over the cutoff value. This is for the distance matrix initialization part of the algorithm, which computes all neighborhood relations (Definition 1). This is the most time-consuming part of the algorithm. Next, the clusters are enumerated. The first cluster is the one containing the structure with the maximum number of neighbors (Definition 2) given the cutoff value. This structure is called the cluster center (Definition 3). This cluster is then removed from the initial set to be clustered, creating the remaining set. Subsequent clusters are found by iterating the procedure until the remaining set is empty. Exact clustering is an unsupervised clustering algorithm.

The distance matrix used during exact clustering can be initialized naively by computing RMSD for each distinct decoy pair (Algorithm 2). For n decoys, it will require the computation of $n(n-1)/2$ RMSDs.

Initializing the distance matrix in an efficient manner is the key part of our approach. Knowing if two decoys are neighbors at a given cutoff is referred to as 'decidability' (Definition 4). For each decoy pair, the algorithm does not always need the exact RMSD. In many cases, knowing either the lower or the upper bound of the distance range is enough to satisfy 'decidability'. We borrowed the lower and upper bounds of C_αRMSD from Calibur (Li and Ng, 2010) and SCUD (Li and Zhou, 2005), respectively. We call 'insertion' of information into the distance matrix as the action of measuring all decoys against a new, randomly chosen reference decoy. We refer to 'propagation' as the use of geometric constraints to propagate the newly acquired information from the previous insertion step into the distance matrix. Only once the distance matrix is fully decided, the algorithm can continue to the cluster enumeration step. We define entropy of the distance matrix as the number of decoy pairs that are still undecided (Definition 5). When we refer to the speed of insertion or propagation (speed_i, speed_p), it means the decrease of entropy during insertion or propagation ($\delta_{\text{entropy}_i}$, $\delta_{\text{entropy}_p}$) divided by the time elapsed during the corresponding step (δ_{t_i}, δ_{t_p}).

Entropy decrease speeds during the insertion and propagation steps are therefore defined as: $\text{speed}_i = \frac{\delta_{\text{entropy}_i}}{\delta_{t_i}}$; $\text{speed}_p = \frac{\delta_{\text{entropy}_p}}{\delta_{t_p}}$.

A crucial part of our algorithm is the propagation of distances which is as follows:

- given three decoys A, B and C (assuming A was randomly chosen to be the new reference decoy);
- we measure AB and AC exactly;
- we subsequently sort these distances in increasing order (let us assume $AB \leq AC$); and
- finally, we can deduce an approximation geometrically for BC, which is: $AC - AB \leq BC \leq AC + AB$.

[3]Sometimes referred to as 'clustering threshold' or even 'cluster cutoff'.

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

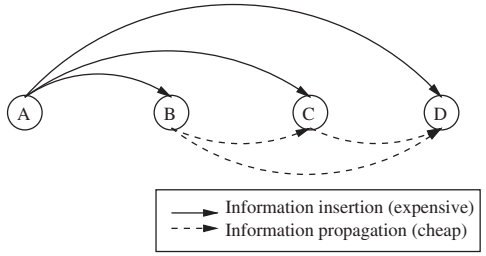


Fig. 1. Generalization of the insert then propagate procedures to four decoys. A is the reference decoy. The distances AB, AC and AD are exactly measured while BC, CD and BD are easily approximated using ranges.

In our approach, this propagation procedure is generalized to any number of decoys (Fig. 1). There are also some optimizations to cut into the $O(n^2)$ complexity of propagation: once distances to the last chosen reference are sorted, it is possible to detect early that processing more distances will not introduce further information into the distance matrix. In this way, all the information that could be exploited from the previous insertion step is used and no time is lost in overexploitation. To guarantee our initialization is faster than the naive method, we monitor at run-time the entropy decrease speed during insertion and propagation steps. Once entropy decrease during insertion become faster than during propagation, our algorithm falls back to a 'lazy' version of the naive distance matrix initialization. For each yet undecided pair, this 'lazy' completion of the distance matrix initialization sequentially tries lower bound, then upper bound and finally resolves to RMSD in case none of the previous trials removed undecidability. Our approach to accelerate initialization of the distance matrix could be used by any other algorithm using a metric distance.

Hereafter, the formal definitions and algorithms previously introduced in prose are given.

2.2 Definitions and algorithms

DEFINITION 1. Neighborhood relation between decoys x and y at distance d:
$(x,y) \in \{decoys\}, d \in \mathbb{R}_{>0}$
$neighbor(x,y,d) = \top \Leftrightarrow rmsd(x,y) <= d$

DEFINITION 2. Number of neighbors for decoy x from cluster c at distance d: $d \in \mathbb{R}_{>0}, c \in \{clusters(\{decoys\},d)\}, x \in c$
$nb\_neighbors(x,c,d) = |\{\forall y \in c \,|\, y \neq x \wedge neighbor(x,y,d)\}|$

DEFINITION 3. Cluster center property for decoy x from cluster c:
$c \in \{clusters(\{decoys\},d)\}, (x,y) \in c$
$center(c,x) = \top \Leftrightarrow \nexists y \,|\, (y \neq x \wedge$
$nb\_neighbors(y,c,d) > nb\_neighbors(x,c,d))$

DEFINITION 4. Decidability criterion for the decoy pair (x,y) at distance d:
$(x,y) \in \{decoys\}, d \in \mathbb{R}_{>0}$
$(rmsd(x,y) \in [r_{min}; r_{max}]) \wedge (0 <= r_{min} <= r_{max})$
$decidable(x,y,d) \Leftrightarrow (r_{min} > d) \vee (r_{max} <= d)$

DEFINITION 5. Entropy for a set of decoys at distance d:
$(x,y) \in \{decoys\}, x \neq y, d \in \mathbb{R}_{>0}$
$entropy(d, \{decoys\}) = |(x,y) \,|\, \neg decidable(x,y,d)|$

It is interesting to note from Definition 3 that there can be several cluster center candidates, if they have an equal number of neighbors at cutoff distance d. The policy to choose the cluster center out of several equally ranked candidates is responsible for the instability of the algorithm. We refer to these equally possible cluster centers as 'pole position centers'. Our implementation has a 'stable' option. It allows to display the pole position centers for the biggest cluster. Note that a cluster center in our definition is not a centroid (i.e. not an average of all the cluster members as is usually understood in the clustering literature). A cluster center in our approach is an actual cluster member (sometimes referred to as a 'clustroid'), contrary to SPICKER which builds cluster representative structures by averaging all cluster members.

Our software implementation offers two ways to choose a cutoff value. The first method is directly via a user-specified value. The second method is semiautomatic. We used the latter during our experiments as it adapts automatically the threshold value to the spread of the decoy set, using a percentage parameter P provided by the user. In semiautomatic mode, Durandal will randomly sample pairwise RMSDs, and use the top $P\%$ sampled one as the cutoff value. The sample is grown iteratively by adding groups of 100 randomly chosen pairwise RMSDs out of the decoy set, to an initially empty set until the median of this set is stabilized.

Algorithm 1 Exact clustering at cutoff d

```
s ← {decoys}
dm ← init_distance_matrix(s,d)
while s ≠ ∅ do
  bc ← biggest_cluster(s,dm)
  output(bc)
  s ← s \ bc
end while
```

Algorithm 2 Naive distance matrix initialization

```
s1 ← {decoys}
s2 ← s1
n ← |s1|
dm ← distance_matrix(n*(n−1)/2, s1)
for x in s1 do
  s2 ← s2 \ x
  for y in s2 do
    dm[index(x)][index(y)] ← rmsd(x,y)
  end for
end for
```

3 RESULTS

3.1 Accuracy

To test the usefulness of our approach, we compared it against published results of SPICKER on I-TASSER decoys (Wu *et al.*, 2007). We selected out of the full set 40 different protein targets. To be included in our study, a decoy set need to contain at least one good quality decoy (i.e. <4 Å C_αRMSD to native). Each time, our software was run using semiautomatic threshold finding with $P = 5\%$ on the full decoy set (no shrinking applied). Results are shown in Figure 2. In each histogram, the first bar is the biggest cluster center discovered by Durandal while the second bar is the decoy nearest to the biggest cluster centroid computed by SPICKER ['closc' files in the decoy set from Wu *et al.* (2007)]. The best energy decoy as well as the nearest to native decoy from each set are shown as reference lines; clustering is useful only if it can pick decoys better than energy. We also tried $P = 3\%$ and $P = 10\%$ as the semiautomatic threshold finding parameter to verify the stability of our method. The maximal variation observed when using these different P-values in terms of average C_αRMSD to native was only around 1.2%.

In our tests, in 25 out of 40 cases the biggest cluster center found by Durandal was nearer to native compared with the best energy decoy (20 out of 40 cases for SPICKER). On average, when comparing the chosen decoy from each method to the

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

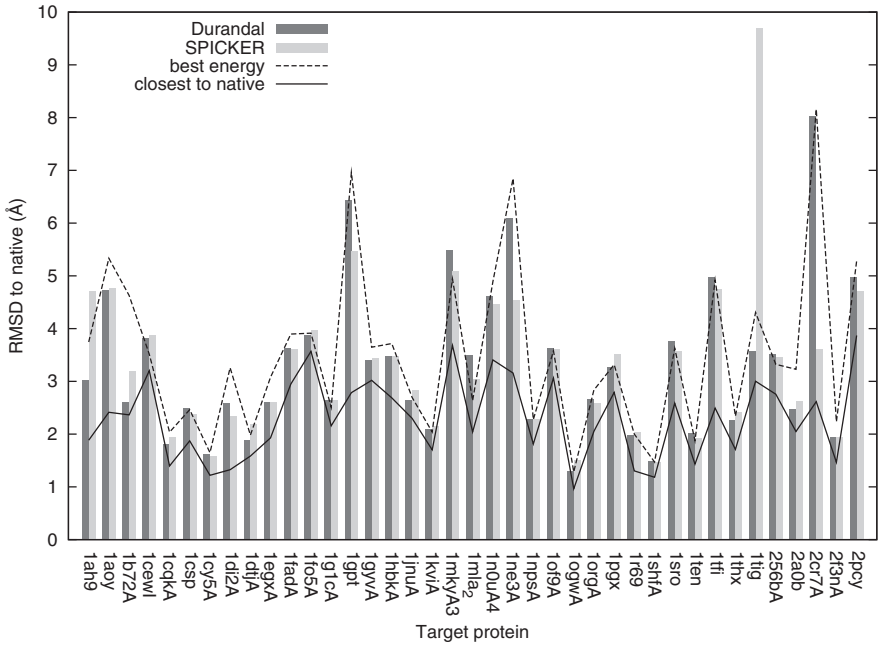


Fig. 2. Clustering good-quality decoy sets from I-TASSER. The y-axis is the C_αRMSD to the native structure. The x-axis is the protein sequence identifier used in I-TASSER, which is a combination of PDB code and chain letter. For each target, Durandal and SPICKER's choices are shown as histogram bars. The best energy decoy and the closest to native decoy in each set are shown as lines.

native structure, Durandal and SPICKER perform almost identically (3.35 Å C_αRMSD and 3.37 Å C_αRMSD, respectively).

3.2 Efficiency

Using geometric constraints propagation, our method allows to exploit efficiently the few C_αRMSD that are computed during the insert-propagate phase (Fig. 3). Also, the speed of entropy decrease criterion allows to fall back to lazy completion at the optimal moment (Fig. 4).

In order to evaluate the speed gained with our method, we measured its execution times on some I-TASSER decoys while varying set size. We also tested our method on the 'semfold' decoy set (Samudrala and Levitt, 2002) (Downloaded from http://dd.compbio.washington.edu/). We compared elapsed times to that of Calibur, which is the fastest software we know that is capable of performing the same task. We also compared elapsed times to that of SCUD which is an approximate method. Calibur was extensively tested against SPICKER and shown to be significantly faster when working on large decoy sets (Li and Ng, 2010). Hence, we did not redo a comparison in speed with SPICKER.

All software were compiled using the highest optimization level (-O3) of their needed compiler. Durandal and Calibur are using the same routines to compute C_αRMSD. All experiments were performed using exactly uniform computer cluster nodes. The PAR (Berenger *et al.*, 2010) software (http://download.savannah.gnu.org/releases/par/par.tgz) was used to parallelize preparation of some input data, experiments, results analysis and obtain a shorter experiment turnaround with all softwares. On each dataset, every software was run five times with the same parameters. The averaged real-time elapsed as reported by the Linux 'time' command was retained. SCUD's run-time does

Algorithm 3 Durandal distance matrix initialization at cutoff d (comments are enclosed between braces).

$s_1 \leftarrow \{decoys\}$
$n \leftarrow |s_1|$
$dm \leftarrow distance\_matrix(n*(n-1)/2, s_1)$
$u_1 \leftarrow \emptyset$
repeat
  $ref \leftarrow random\_choice(s_1)$
  $s_1 \leftarrow s_1 \setminus ref$
  $speed_i \leftarrow insert(ref, dm)$ {insert new information}
  $speed_p \leftarrow propagate(ref, dm)$ {propagate it}
  $u_1 \leftarrow undecided\_pairs(dm)$
until $(speed_i > speed_p) \vee (u_1 = \emptyset)$
$u_2 \leftarrow u_1$
for x in u_1 **do**
  {lazy completion of the initialization}
  $u_2 \leftarrow u_2 \setminus x$
  for y in u_2 **do**
    $r_{min} \leftarrow lower\_bound\_rmsd(x, y)$
    $r \leftarrow [r_{min}; \infty]$
    if $\neg decidable(r, d)$ **then**
      $r_{max} \leftarrow upper\_bound\_rmsd(x, y)$
      $r \leftarrow [r_{min}; r_{max}]$
      if $\neg decidable(r, d)$ **then**
        $rms \leftarrow rmsd(x, y)$
        $r \leftarrow [rms; rms]$
      end if
    end if
    $dm[index(x)][index(y)] \leftarrow r$
  end for
end for

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

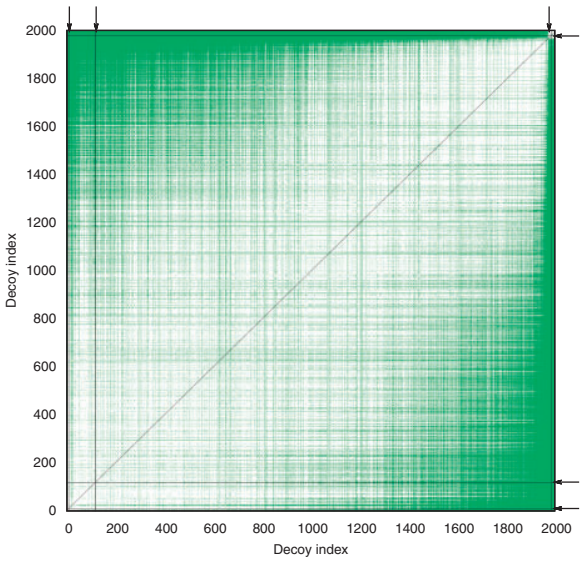


Fig. 3. Colorized snapshot of the distance matrix after three insertion and three propagation steps (Algorithm 3). While having measured only 0.35% of all the possible C_αRMSD pairs, our method completed 34.25% of the distance matrix initialization. White points are yet undecided decoy pairs. Green points are decided pairs that were estimated during propagation steps. Each black line is composed of decided pairs sharing the same reference decoy, exactly measured using C_αRMSD during an insertion step. As the black lines from the three insertion steps plus their three symmetry mates are very thin, their positions are indicated by arrows at the upper and right sides of the picture. The black diagonal is the symmetry axis and also the not computed zero distance to self of each decoy. Decoys are sorted in the order of increasing C_αRMSD to the decoy at index 0 for display purpose. The cutoff was 0.75 Å for 2k decoys of protein target 1aoy.

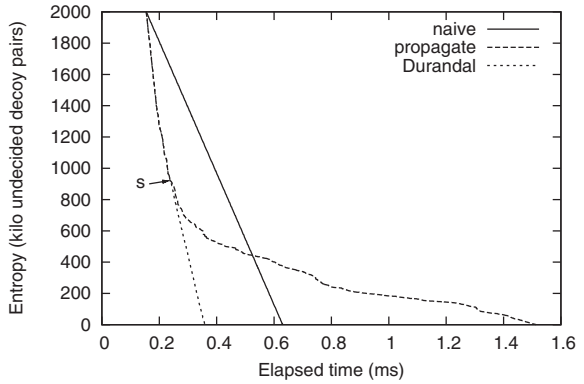


Fig. 4. Different algorithms initializing the distance matrix at cutoff 0.75 Å on 2k decoys for protein target 1aoy. Entropy reaching zero is the terminating condition of the distance matrix initialization (a mandatory step prior to enumerating clusters). On the Durandal plot, the arrow labeled s indicates a switch in distance matrix initialization strategy from insert-propagate to lazy-completion (Algorithm 3).

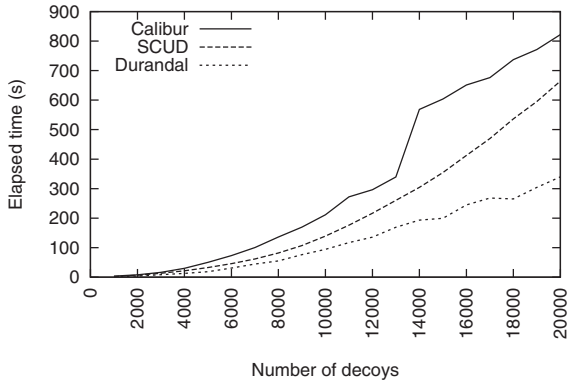


Fig. 5. Speed comparison between Calibur, SCUD and Durandal on 1shfA decoys (1.5 Å cutoff). For clarity, only one slice in the 3D landscape created by varying the cutoff from 0.5 to 8 Å by 0.5 Å increment is shown. When considering the full cutoff range, Durandal is faster than Calibur in 98% of all cases. SCUD results were computed only for this slice.

not vary as a function of the clustering threshold. However, the Calibur and Durandal run-times vary in a difficult to predict manner as a function of the clustering threshold and the distribution of decoys in the set. Durandal's run-time is also influenced by which references are used. All methods' speeds are influenced by the number of decoys in the set and the number of residues. Calibur was run without outlier filtering (to ensure exact clustering of the whole set is performed, as Durandal does). During Durandal runs, random reference selection was used to avoid input order bias. If references being used do not differ much from each other because they are chosen sequentially from the input files list and this list is somewhat ordered, it would penalize our algorithm. Results of comparison in speed with Calibur and SCUD while varying the decoy set size are shown in Figure 5. Accelerations observed on the 'semfold' decoy set are shown in Table 1. As a test to verify that both Calibur and Durandal implement correctly the exact clustering algorithm, some output files were randomly chosen and compared, showing a consistent 100% cluster similarity among the three biggest clusters (smaller clusters were not saved during experiments as Calibur outputs only top three by default).

In Figure 5, the bump in Calibur run-times observed for set sizes larger than 13k decoys is due to a change of storage data structure at run-time (a default Calibur behavior that the user has no control over). In Figure 5, the average acceleration rate obtained by Durandal compared with Calibur is 2.53 and 1.74 compared with SCUD. In Table 1, our method is shown to be consistently faster than Calibur and can even outperform SCUD in some cases.

4 DISCUSSION

Distance matrix computation is the most time consuming part of exact clustering. It is interesting to look at how the same problem was approached in two different ways by Calibur and Durandal. Calibur's algorithm grows clusters stepwise in an online manner. Calibur's focus is targeted at grouping together proximate decoys so the problem is looked from a spatial organization viewpoint. Durandal computes clusters off-line, once enough information is known regarding distances of all decoy pairs. Distance information are inserted into the distance matrix using methods to maximize

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

Table 1. Acceleration rate obtained by Durandal compared with SCUD and Calibur on the 'semfold' decoy set

Target (decoys, residues)	Cutoff (Å)	SCUD	Calibur
1pgb	5	1.21	1.69
(11 280, 56)	6	0.9	1.49
	7	0.75	1.61
	8	0.8	1.71
1e68	4	2.08	1.75
(11 361, 70)	5	1.54	1.44
	6	1.15	1.38
	7	0.95	1.61
1ctf	4	1.43	1.61
(11 400, 68)	5	1.38	1.57
	6	1.08	1.47
	7	0.86	1.52
1eh2	6	1.43	1.35
(11 440, 95)	7	1.01	1.06
	8	0.85	1.28
	9	0.84	1.63
1nkl	5	1.73	1.42
(11 660, 78)	6	1.23	1.17
	7	0.95	1.3
	8	0.81	1.51
1khm	5	1.99	3.22
(21 080, 73)	6	1.39	2.74
	7	1.01	2.84
	8	0.85	3.4

The lower clustering threshold for each target was chosen so that the first cluster is statistically significant. Calibur and Durandal are exact methods while SCUD is approximate. The acceleration rates labeled 'SCUD' or 'Calibur' in the table are the total run-time of 'SCUD' or 'Calibur' divided by the total run-time of 'Durandal'.

the speed of the procedure. Durandal attacks the problem using an information-centered approach. One of the consequences of these different strategies is that Calibur does not fully exploit some of the expensive methods to compute information it collects. In Calibur, some previously computed distances or their bounds are used to avoid computing some more distances, but this information does not percolate through all clusters. Whereas in Durandal, as long as inserting then propagating new distance information is the fastest strategy, information is exploited to the maximum.

Both SPICKER and Durandal implement the same clustering algorithm. However, some slight differences remain. Both methods use different automatic threshold-finding strategies, decoy selection technique and energy filtering. Concerning threshold-finding strategy, SPICKER follows a cutoff refinement algorithm (Zhang and Skolnick, 2004a) while Durandal picks a value after some quick sampling of the decoy set. Durandal's way is close to Calibur's default threshold-finding strategy. For decoy selection, SPICKER selects the decoy nearest to the cluster centroid while Durandal selects the cluster center as understood in Definition 3. Moreover, Durandal clusters full decoy sets without applying additional energy filtering, whereas SPICKER reduces the decoy set to 13k decoys maximum using an energy criterion. When the energy function is inaccurate, Durandal may benefit more from the averaging effect of clustering.

We have addressed the problem of clustering speed to a large extent with Durandal. However, the memory requirement issue has not been dealt with. The memory requirement grows in the same order as that of the naive algorithm. This is an important problem. Clustering acceleration can be tackled using geometric constraints as we did. If the distance measure being used is not a metric, our approach cannot be applied to speed up the process by reducing the number of calls to the distance function. The memory problem is not easy to address; if the distance matrix were stored on disk, the overall clustering procedure would become an order of magnitude slower due to the higher disk access latency compared with memory. Algorithms avoiding the memory size problem should work under the constraint: being able to cluster without requiring that the full distance matrix is available in memory. In this aspect, despite Calibur's speed being far from optimal, it is worthwhile noticing that it is quite thrifty regarding memory consumption.

Concerning the optimum choice of reference decoys in Durandal, previous work employing a metric distance in the problem of best-match search showed that there is an optimal heuristic to choose reference points (Shapiro, 1977). Reference points should be away from cluster centers. However, the exact distance away, as well as the number of reference points to use is unclear. Concerning the number of references to consider, our algorithm uses an entropy decrease speed criterion to detect if we are starting to use too many of them in order to trigger a fallback from the insert-propagate step to the lazy-completion one. Durandal uses chance in its reference point choice policy. Based on our experience using Durandal, the speed criterion combined with random reference selection creates an efficient and simple heuristic.

Some distributed algorithm could push even further the scalability of clustering large decoy sets, both in terms of speed and memory requirements. In the distributed case, the problem would probably be no longer just about fast clustering where our algorithm could be used. The challenge may become how to efficiently merge and store intermediate clustering results from partitions of a whole decoy set. Possibly, some small overlapping in the partitions could accelerate the procedure by merging first reference structures from both sets. Afterwards, many distance measures would be avoided. If the user is not interested in parallelizing computations, the parallel version of the program could still be used to overcome the memory barrier by running it sequentially on a single computer. Partitioning the initial set, and then clustering each subset before incrementally merging results seems a feasible approach.

We hope the method we proposed will help the protein folding community in the decoy identification step of their protocols. The acceleration technique we detailed may be of interest to other scientific or engineering fields and may be adapted to other clustering algorithms.

ACKNOWLEDGEMENTS

We wish to thank RIKEN, Japan, for an allocation of computing resources on the RIKEN Integrated Cluster of Clusters (RICC) system. We are largely indebted to protein folding researchers who make publicly available full decoy sets.

Funding: Initiative Research Unit program from RIKEN, Japan.

Conflict of Interest: none declared.

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

REFERENCES

Anfinsen,C.B. (1973) Principles that govern the folding of protein chains. *Science*, **181**, 223–230.

Berenger,F. *et al.* (2010) PAR: a PARallel and distributed job crusher. *Bioinformatics*, **26**, 2918–2919.

Betancourt,M.R. and Skolnick,J. (2001) Finding the needle in a haystack: educing native folds from ambiguous ab initio protein structure predictions. *J. Comput. Chem.*, **22**, 339–353.

Bonneau,R. *et al.* (2001) Rosetta in casp4: Progress in ab initio protein structure prediction. *Proteins Struct. Funct. Bioinformatics*, **45**, 119–126.

Das,R. and Baker,D. (2008) Macromolecular modeling with rosetta. *Annu. Rev. Biochem.*, **77**, 363–382.

Gront,D. and Kolinski,A. (2005) Hcpm–program for hierarchical clustering of protein models. *Bioinformatics*, **21**, 3179–3180.

Gront,D. *et al.* (2005) Exploring protein energy landscapes with hierarchical clustering. *Int. J. Quant. Chem.*, **105**, 828–830.

Hastie,T. *et al.* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. 2nd edn. corr. 3rd printing edition.

Ishida,T. *et al.* (2003) Development of an ab initio protein structure prediction system able. *Genome Inform.*, **14**, 228–237.

Jain,A.K. *et al.* (1999) Data clustering: A review. *ACM Comput. Surv.*, **31**, 264–323.

Kedem,K. *et al.* (1999) Unit-vector rms (urms) as a tool to analyze molecular dynamics trajectories. *Proteins*, **37**, 554–564.

Kmiecik,S. and Kolinski,A. (2008) Folding pathway of the b1 domain of protein g explored by multiscale modeling. *Biophys. J.*, **94**, 726–736.

Kohonen,T. (1998) The self-organizing map. *Neurocomputing*, **21**, 1–6.

Li,H. and Zhou,Y. (2005) Scud: Fast structure clustering of decoys using reference state to remove overall rotation. *J. Comput. Chem.*, **26**, 1189–1192.

Li,H. (2006) A model of local-minima distribution on conformational space and its application to protein structure prediction. *Proteins Struct. Funct. Bioinformatics*, **64**, 985–991.

Li,S. and Ng,Y. (2010) Calibur: a tool for clustering large numbers of protein decoys. *BMC Bioinformatics*, **11**, 25.

Raman,S. *et al.* (2009) Structure prediction for casp8 with all-atom refinement using rosetta. *Proteins Struct. Funct. Bioinformatics*, **77**, 89–99.

Samudrala,R. and Levitt,M. (2002) A comprehensive analysis of 40 blind protein structure predictions. *BMC Struct. Biol.*, **2**, 3.

Shapiro,M. (1977) The choice of reference points in best-match file searching. *Commun. ACM*, **20**, 339–343.

Shortle,D. *et al.* (1998) Clustering of low-energy conformations near the native structures of small proteins. *Proc. Natl Acad. Sci. USA*, **95**, 11158–11162.

Skolnick,J. (2006) In quest of an empirical potential for protein structure prediction. *Curr. Opin. Struct. Biol.*, **16**, 166–171.

Steipe,B. (2002) A revised proof of the metric properties of optimally superimposed vector sets. *Acta Crystallogr. Sect. A*, **58**, 506.

Wu,S. *et al.* (2007) Ab initio modeling of small proteins by iterative tasser simulations. *BMC Biol.*, **5**, 17.

Zhang,Y. and Skolnick,J. (2004a) Spicker: a clustering approach to identify near-native protein folds. *J. Comput. Chem.*, **25**, 865–871.

Zhang,Y. and Skolnick,J. (2004b) Tertiary structure predictions on a comprehensive benchmark of medium to large size proteins. *Biophys. J.*, **87**, 2647 – 2655.

Zhang,Y. *et al.* (2005) Tasser: an automated method for the prediction of protein tertiary structures in casp6. *Proteins Struct. Funct. Bioinformatics*, **61**, 91–98.

Downloaded from http://bioinformatics.oxfordjournals.org/ at ENS BIOLOGIE on November 13, 2014

3.3 Comparaison électrostatique d'une protéine avec une petite molécule

L'un des principes de base des complexes protéine-ligand est que la molécule active (le ligand) est complémentaire à sa protéine réceptrice. Si l'on considère que la protéine est une serrure, alors la molécule active est l'une des clés possibles. Ce principe implique que des molécules actives sur un même récepteur peuvent partager des traits communs.

Nous avons développé un logiciel afin d'investiguer si la similarité dans l'espace électrostatique peut être utilisée pour la découverte d'inhibiteurs d'interactions entre protéines.

Notre logiciel mesure la similarité de potentiels électrostatiques au voisinage d'une molécule et d'une protéine ligante. Le logiciel est appelé EleKit, pour « Electrostatic Kit ». L'analyse de triplets existants dans la littérature (petite molécule, protéine ligante et protéine réceptrice) montre que les petites molécules inhibitrices d'une interaction entre protéines ressemblent souvent aux protéines avec lesquelles elles entrent en compétition. Cette observation est surtout vraie pour les protéines avec les sites actifs les plus polaires. EleKit est un outil de mesure qui peut être utile lors de la conception d'une molécule destinée à perturber une interaction protéine-protéine.

OPEN ACCESS Freely available online

PLOS ONE

Electrostatic Similarities between Protein and Small Molecule Ligands Facilitate the Design of Protein-Protein Interaction Inhibitors

Arnout Voet[•], **Francois Berenger**[•], **Kam Y. J. Zhang***

Zhang Initiative Research Unit, Institute Laboratories, RIKEN, Wako, Saitama, Japan

Abstract

One of the underlying principles in drug discovery is that a biologically active compound is complimentary in shape and molecular recognition features to its receptor. This principle infers that molecules binding to the same receptor may share some common features. Here, we have investigated whether the electrostatic similarity can be used for the discovery of small molecule protein-protein interaction inhibitors (SMPPIIs). We have developed a method that can be used to evaluate the similarity of electrostatic potentials between small molecules and known protein ligands. This method was implemented in a software called EleKit. Analyses of all available (at the time of research) SMPPII structures indicate that SMPPIIs bear some similarities of electrostatic potential with the ligand proteins of the same receptor. This is especially true for the more polar SMPPIIs. Retrospective analysis of several successful SMPPIIs has shown the applicability of EleKit in the design of new SMPPIIs.

Citation: Voet A, Berenger F, Zhang KYJ (2013) Electrostatic Similarities between Protein and Small Molecule Ligands Facilitate the Design of Protein-Protein Interaction Inhibitors. PLoS ONE 8(10): e75762. doi:10.1371/journal.pone.0075762

Editor: Andreas Hofmann, Griffith University, Australia

Received February 20, 2013; **Accepted** August 19, 2013; **Published** October 10, 2013

Copyright: © 2013 Voet et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Funding: This work was supported by the JSPS and the Initiative Research Unit program from RIKEN, Japan. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: kamzhang@riken.jp

• These authors contributed equally to this work.

Introduction

With the advent of the 'omics' era, it has become clear that most proteins do not act in solitude but depend on Protein-Protein Interactions (PPIs) to exert their biological function. It has been estimated that the number of PPIs in humans ranges from ~130,000 [1] to ~650,000 [2] and these PPIs are crucial for the regulation of many biological processes. PPIs are often involved in processes associated with diseases, therefore targeting PPIs with small molecule PPI inhibitors (SMPPIIs) opens a pipeline for the development of novel drug classes against a variety of diseases.

While many small molecule drugs targeting enzymes, nuclear receptors, ion channels and G-protein coupled receptors have been developed, the number of reported successes in the discovery of SMPPIIs remains fairly low. As a matter of fact, PPIs were once thought to be high hanging fruits for drug discovery [3]. PPIs were even considered to be undruggable, mostly because of their relative flat but extensive interfaces [4]. Though initially thought to be undruggable, an increasing number of SMPPIIs have been reported in recent years [5]. However, the number of deposited 3D SMPPII receptor complex structures remain far more limited than the number of reported successful cases. This hinders the understanding of their mechanism of action and chemical space properties [6]. Commonly used methods for screening are computational docking [7] and pharmacophore-based screening [8]. It was observed that the crucial interactions between a protein ligand and its protein receptor are often similar to those between the SMPPII and the protein receptor [9,10]. Thus, the PPI interface can be used to create a pharmacophore query to screen for small molecule ligands [11,12].

Another approach is to exploit the principle of electrostatic complementarity in molecular recognition. Next to steric complementarity, electrostatics are one of the main driving forces involved in molecular recognition [13]. Despite the complex biophysical nature of the electrostatic potential, calculations for macromolecular systems are nowadays tractable [14,15].

Electrostatics are known to play a key role in protein-DNA [16], protein-protein [17] and protein-substrate [13] recognitions. Given the importance of electrostatics for the molecular recognition event, electrostatics have been used to study protein similarity [18–20] and the nature of protein-protein interactions [17,21–24]. More specifically, the electrostatic complementarity between protein-protein interfaces has long been a subject of investigation [22,23]. Using the correlation of electrostatic potentials as a quantitative measure, the electrostatic complementarity between PPI interfaces has been demonstrated [17,24]. Other studies focused on the conservation of the electrostatic potentials through evolution [25] and its role in molecular association kinetics [26].

It is generally accepted that there is a high degree of complementarity in shape and electrostatics between a ligand and its receptor. This implies that molecules with similar shape and electrostatic properties may bind to the same receptor. This principle has been used to identify small molecule inhibitors similar to natural substrates or known inhibitors by screening for compounds with similar shape, volume and electrostatics [27–30].

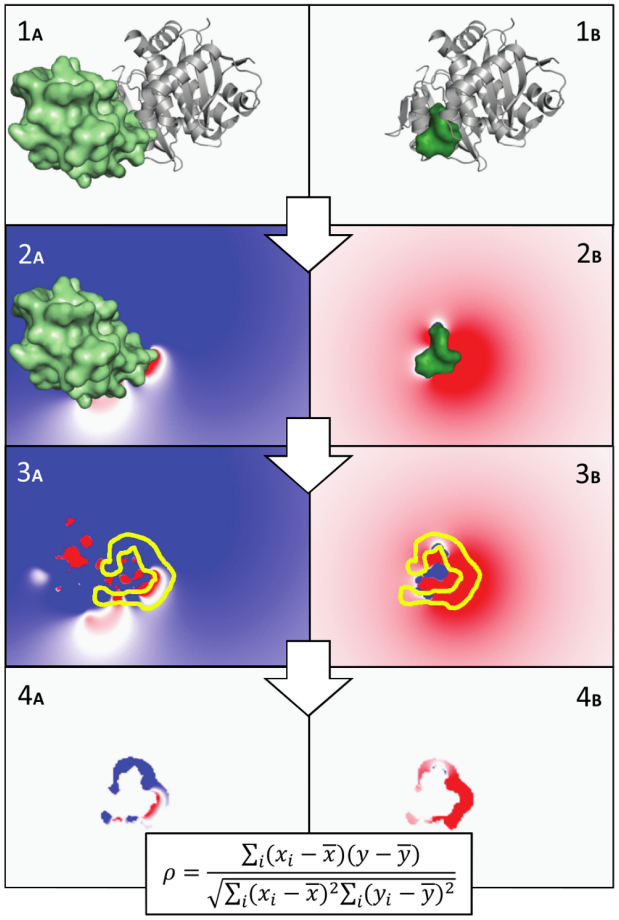


Figure 1. Overview of EleKit applied to PDB codes 2B4J (1_A) and 3LPU (1_B). The ligand protein (L_P) is shown as a green surface in 1_A and 2_A. The ligand small molecule (L_{SM}) is shown as a smaller green surface in 1_B and 2_B. The receptor protein (R_P) is shown as a gray cartoon in 1_A and 1_B. and are placed on (1_A and 1_B). The electrostatic potentials of and are calculated and stored in distinct grids (2_A and 2_B). Then, a mask is created to select the solvent region near the interface (3_A and 3_B). Finally, the similarity between electrostatic potentials of and over this region is calculated using the Spearman rank correlation coefficient (4_A and 4_B).
doi:10.1371/journal.pone.0075762.g001

An SMPPII cannot occupy the same shape and volume as its much bigger protein-ligand counterpart. However, it can still be assumed that there is some local electrostatic potential similarity between an SMPPII and a ligand protein, since they recognize the same binding site on the receptor. A recent example of the usefulness of taking electrostatic potential similarity into account while designing an SMPPII can be found in the work of Cavalluzo *et al.* [31], where an SMPPII was designed *de novo* by including electrostatic similarity. This success has motivated our effort to systematically investigate the complementarity in electrostatic potential between small molecules and protein ligands binding to the same protein receptor, and its potential use to assist in the rational design of SMPPIIs. For this purpose, a tool named EleKit was developed.

Table 1. APBS commands used for a protein.

read	
	mol pqr ligandprotein.pqr
end	
elec	
	mg-auto
	dime 161 193 161
	cglen 105.5210 127.4643 91.1458
	fglen 82.0712 94.9790 73.6152
	cgcent -6.603267 -6.904766 -18.393622
	fgcent -6.603267 -6.904766 -18.393622
	mol 1
	lpbe
	bcfl sdh
	pdie 2.0000
	sdie 78.5400
	srfm smol
	chgm spl2
	sdens 10.00
	srad 1.40
	swin 0.30
	temp 298.15
	calcenergy no
	calcforce no
	write smol dx ligandprotein.ms
	write pot dx ligandprotein
end	
quit	

For a small molecule, the input and output file names would differ. Parameters were determined with pdb2pqr.py. Grid-related parameters vary upon each case (dime, cglen, fglen, cgcent and fgcent).
doi:10.1371/journal.pone.0075762.t001

Methods

To compute the partial charges and electrostatic potentials, EleKit builds upon PDB2PQR [32] and APBS [15]. EleKit requires two sets of complex structures in order to calculate the electrostatic similarity between a protein ligand and a small molecule ligand: (i) the PPI complex of the protein-ligand (L_P) with the protein-receptor (R_P) and (ii) a small molecule ligand (L_{SM}) in its predicted or experimentally determined conformation on the protein-receptor (R_P).

The EleKit method is shown schematically in figure 1. First, the electrostatic potentials around and are computed using APBS (parameters listed in table 1) and stored in 3D grids. Since only the area where and intersect is most likely to be relevant for molecular recognition, a bit mask is created on the electrostatic potential grids (figure 2). The goal of this mask is to take into account only those points in space that are not only in the solvent region around and but also near the interface atoms of R_P. To create this mask, a distance cutoff is needed. This distance is used when dilating (a morphological mathematical operation) the molecular surface. Based on the hydrogen bond length (~2.5Å) and the facts that enough points are needed for correlation and that the local similarity is our focus, a cutoff value ranging from 1.4 Å to 3.5 Å seems reasonable. All experiments reported in this study were

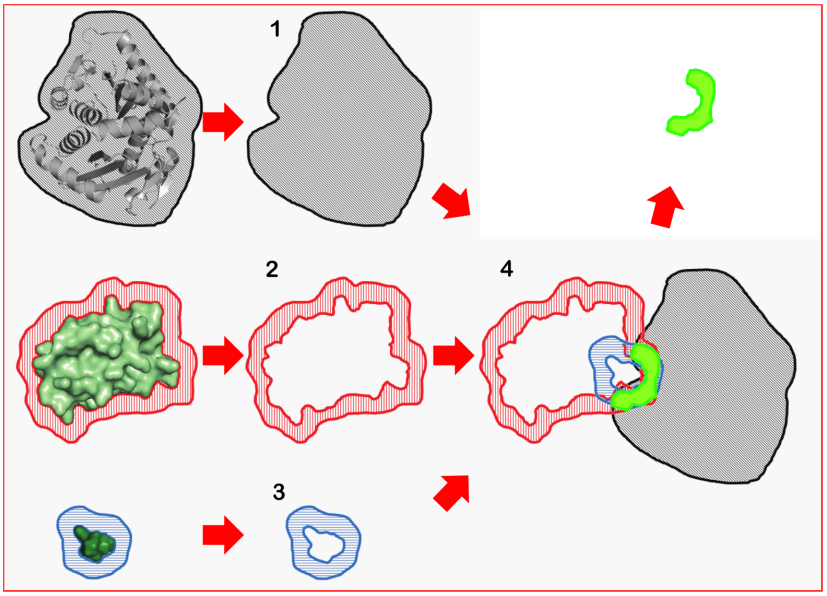


Figure 2. Overview of the bit-mask construction in EleKit. (1) a near-or-inside mask of R_P is created, (2) a near-but-not-inside mask of L_P is created, (3) a near-but-not-inside mask of L_{SM} is created, (4) the logical conjunction of the three masks is used to select points to correlate from the electrostatic potentials of L_P and L_{SM}.
doi:10.1371/journal.pone.0075762.g002

performed with an intermediate cutoff value of 2.0 Å. Using 3.0 Å or 4.0 Å would have very little impact on the results (data not shown). Finally, the similarity between electrostatic potentials of and is assessed by correlating values at the grid points within the mask using the Spearman rank-order correlation coefficient (ρ). Additional similarity scores (Carbo index [33], Hodgkin index [34], Pearson's r and a Tanimoto score) are also calculated.

EleKit is written in OCaml [35] (http://biocaml.org) and computations are parallelized by the Parmap library [36]. Experiments were run on Linux computing nodes with 2.4GHz Intel Xeon processors. EleKit takes between ten seconds to two minutes per ligand molecule and can parallelize the computation of several ligands when run on a multi-core machine. EleKit is released as open source and available from the authors' website http://www.riken.jp/zhangiru/software.html.

Results and Discussion

The EleKit method was applied to analyze previously reported cases of SMPPIIs, for which accurate structures of the PPI as well as the SMPPII receptor complex are available in the PDB (table 2). Additionally, the SMPPIIs are required to bind in the PPI interface, allowing for a substantial overlap between the protein ligand and the SMPPII and thus excluding allosteric inhibition mechanisms.

The approach used in EleKit to perform comparison of electrostatic potentials resembles what has been done previously on proteins [18–21]. Analysis of Electrostatic Similarities of Proteins (AESOP) [20], the method of Dlugosz *et al.* [19] and Protein Interaction Property Similarity Analysis (PIPSA) [18,37] also use APBS as their electrostatic computation engine. PIPSA can also use University of Houston Brownian Dynamics [38] (UHBD). While EleKit relies on the Spearman rank-order correlation coefficient (as McCoy *et al.* [17]), PIPSA uses the Hodgkin index [34] to numerically assess the similarity of electrostatic potentials. AESOP uses the Average Normalized Difference [39]. The method of Dlugosz *et al.* [19] approximates the electrostatic potential with spherical harmonics and uses a similarity index specifically designed to compare the obtained rotation-invariant descriptors. EleKit, similarly to several other methods [18,19,37], uses boolean masks to select a region over which electrostatic potentials are compared. All methods vary in the way masks are constructed.

Analysis with EleKit

Electrostatic similarity analysis for these different SMPPII-related structures indicate that several exhibit correlation. In general, correlation between electrostatic potentials of SMPPIIs and electrostatic potentials of the respective ligand proteins are observed (table 2). This is especially true for the SMPPIIs targeting the HDM2:p53, HIV-1 Integrase:LEDGF/p75, Integrin:Fibrinogen, IL2:IL2R and XIAP:smac interactions. The highest similarity between a protein ligand and a small molecule ligand can be observed in the HIV-1 Integrase:LEDGF/p75 and the Integrin:Fibrinogen interactions and their respective inhibitors. In these cases, ρ is on average $\sim$0.52 and $\sim$0.73 respectively (table 2). The origin of these classes of SMPPIIs can be traced back to pharmacophore based discovery of lead compounds designed to mimic the interactions observed at the PPI interface [40,41].

For the inhibitors of the HDM2:p53 interaction, the majority of the inhibitors exhibit electrostatic potential similarity. However, a few show low correlations ($\rho < 0.2$) and in one case even some anti-correlation ($\rho \approx -0.15$). Interestingly, the Tanimoto score shows similarity in all HDM2:p53 cases. The electrostatic potentials between inhibitors and protein ligands in ZipA:FtsZ and VHL:HIF1 still correlate although less strongly than in other cases. These inhibitors are observed to be less active when tested. For inhibitors targeting the XIAP:smac interaction, which originated from peptidomimetic design, some compounds exhibit lower similarity than expected. This can be explained by the divergence of conformations of the receptor protein, since the XIAP:smac complex was solved by NMR while the structures of

Table 2. Analysis of SMPPII electrostatic mimicry.

PPI target	SMPPI complex (observations)	#points	ρ	r	t_2
HDM2:p53	1rv1	5515	0.275	0.263	0.816
(1ycr)	1t4e (g)	5281	0.433	0.355	0.757
	1ttv	4862	-0.152	-0.123	0.588
	3jkz	4384	0.200	0.018	0.566
	3lbj (g)	5159	0.353	0.190	0.611
	3lbk (g)	4681	0.316	0.209	0.883
	3lbl	4129	0.131	-0.015	0.672
	3tu1 (g)	4492	0.589	0.329	0.764
	4dij	4758	0.239	0.111	0.619
IL2:IL2R	1m48 (g)	3186	0.315	0.276	0.658
(1z92)	1m49 (g)	3523	0.758	0.654	0.737
	1py2	3231	-0.011	0.095	0.583
	1pw6 (g)	4119	0.653	0.574	0.678
	1qvn	3819	0.093	0.098	0.626
Integrase:LEDGF/p75	3lpt (g)	3653	0.468	0.437	0.205
(2b4j)	3lpu (g)	3489	0.460	0.464	0.220
	4dmn (g)	3880	0.471	0.534	0.229
	4e1m (g)	3612	0.609	0.601	0.228
	4e1n (g)	3486	0.615	0.591	0.267
Integrin:Fibrinogen	2vdm (g)	6471	0.731	0.751	0.617
(2vdo)	2vc2 (g)	6469	0.736	0.720	0.614
VHL:HIF1 (1lqb)	3zrc (g)	4135	0.123	0.127	0.062
XIAP:smac	2i3i (g)	5979	0.535	0.501	0.650
(1g3f)	1tft (g)	5853	0.377	0.409	0.525
	3eyl (g)	4377	0.321	0.375	0.506
	2jk7 (g)	5446	0.464	0.447	0.631
	3clx	5570	0.055	0.127	0.431
	3cm7	6005	0.227	0.229	0.534
	3hl5	6044	0.263	0.302	0.484
	3mup (g)	6028	0.480	0.404	0.671
	3g76 (g)	5008	0.396	0.391	0.371
	3oz1	5340	0.191	0.184	0.580
ZipA:FtsZ	1s1s (w_i)	4320	0.150	-0.061	0.704
(1f47)	1y2f (w_i, g)	3878	0.365	0.173	0.604
	1y2g (w_i)	3809	-0.318	-0.147	0.335
	1s1j (w_i, g)	2691	0.331	0.088	0.465

ρ: Spearman rank correlation coefficient; r: Pearson linear correlation coefficient; t2: a Tanimoto score (positive or negative electrostatic potential); observations: **g** = good ($\rho > 0.3$), $\mathbf{w_i}$ = weak inhibitor (potency <100 μM in the literature).

doi:10.1371/journal.pone.0075762.t002

XIAP bound to inhibitors were solved by X-ray crystallography. The PPI complex solved by NMR spectroscopy are more difficult to superpose onto the crystal structure conformation obtained for the SMPPII complex. The inhibitors of the IL2:IL2R interaction are well known for binding to the IL2R interface by causing a rotameric change of a phenyl alanine creating a binding pocket [42]. In this case, the PPI interface is only partially covered in a hydrophobic area caused by the induced fit. However, the observed similarity between the ligand protein and the inhibitor mainly originates from the mimicry of the arginine guanidinium group, which is not influenced by conformational changes or induced fit.

There are no significant electrostatic correlations found in the cases of the inhibitors of the Bcl2 family of proteins, the TNFα trimerization and the HPV polymerase. A careful analysis of the structures of these molecules revealed that the SMPPII in these cases is bound after a major reorganization of the receptor protein surface at the PPI interface. For the SMPPIIs bound to the Bcl2

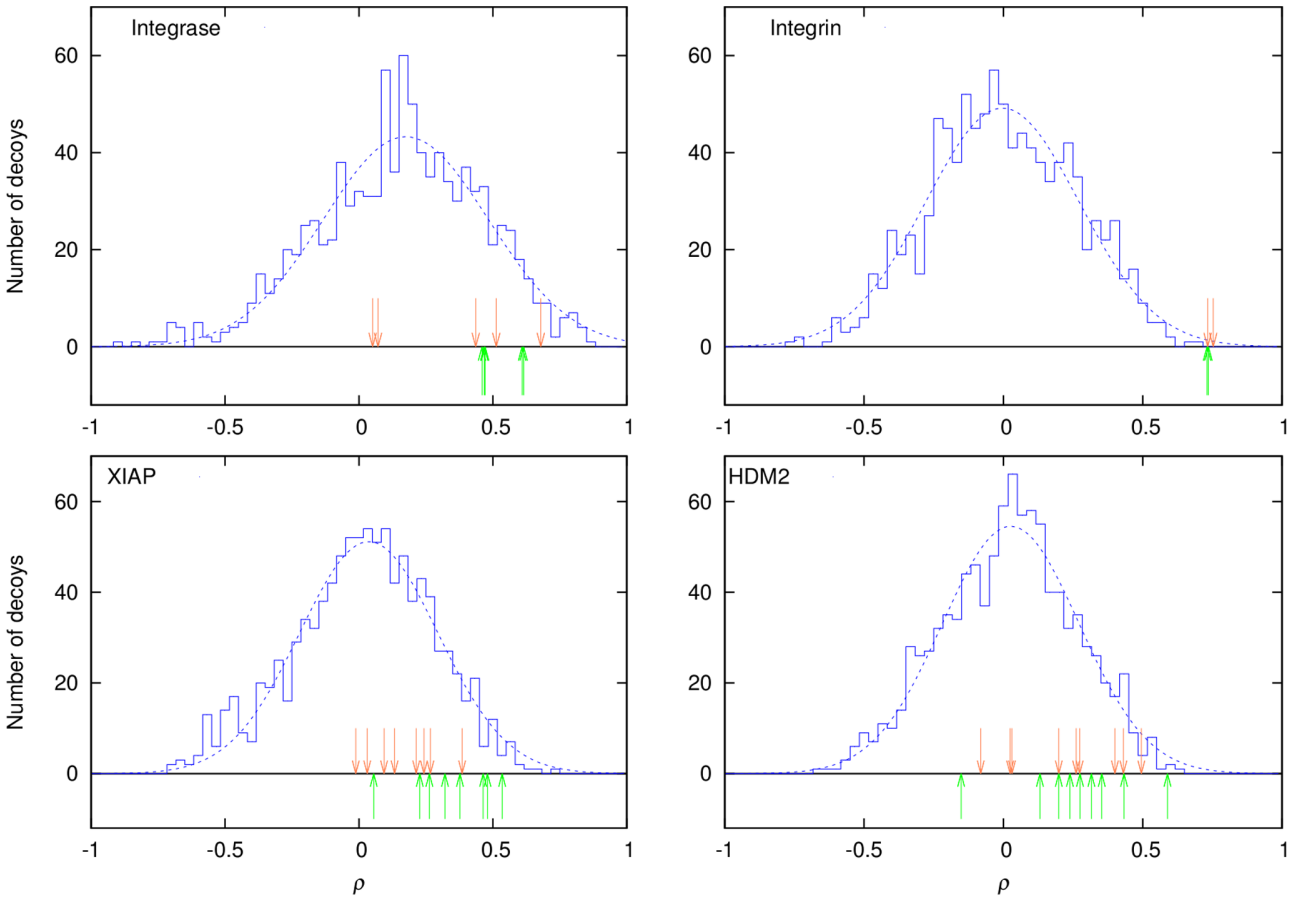


Figure 3. Distribution of Spearman scores (ρ) for active ligands among decoys. EleKit Spearman scores for a population of ligand decoys are shown as a blue histogram. The active ligands at their best docked pose in terms of RMSD to the crystallized active ligand are shown as orange arrows and the crystallized active ligands as green arrows. The dotted line is a Gaussian fitted to the decoys' histogram.
doi:10.1371/journal.pone.0075762.g003

family proteins, there is a major induced fit not only involving side chain atoms, but also including a rearrangement of a single helix, in order to comfortably fit the SMPPII inside the same cleft that was originally occupied by a full and more bulky α-helical protein ligand. The inhibitors of the TNFα and HPV polymerase bind in a pocket at the PPI interface created by the assumption of different side chain orientations with more open conformations. Furthermore, the SMPPIIs that break the E1:E2 interaction of the HPV polymerase act as a dimer. In these cases, the SMPPIIs do not act by mimicking and competing with the ligand protein and no similarity of electrostatic potentials is observed.

EleKit is able to assess electrostatic potential similarity by a variety of measures including ρ, r and a Tanimoto score (table 2). Overall, relying on ρ over r is preferred as it is more robust and does not suffer from uncertainties in interpreting the significance of the observed correlations [43]. The Tanimoto score, which is based on binning the electrostatic potential values as positive or negative does not seem to work in cases where the overall charge of the protein ligand is significantly different from that of the small molecule. An example is the case of the inhibitors of the Integrase:LEDGF/p75 interaction, where the protein ligand has a high positive net charge while the SMPPII has one negatively charged group. The Tanimoto scores are significantly affected and tend to be low. This problem is not observed when using correlation scores. The local shape similarity between the protein ligand and an SMPPII in EleKit is reflected by the number of electrostatic potential grid points being correlated. The more locally similar in shape a ligand protein is to a given SMPPI, the more grid-points will remain in the final mask.

Application of EleKit

To test the utility of EleKit for the post-filtering of results from virtual screening such as docking, four different cases where strong electrostatic similarities between SMPIIs and protein ligands have been observed were selected for further analysis. In each case, a set of 100 diverse decoy compounds that bear similar chemical properties as the active SMPPII were extracted from the purchasable subset of the ZINC database [44] using the procedure described by Huang *et al.* [45,46] to create the Directory of Useful Decoys (DUD).

To position these decoy compounds inside the receptor protein, as if it were a virtual screening experiment, molecular docking was performed using the Molecular Operating Environment (MOE) with the London dG score [47,48]. The receptor protein structure was extracted from the coordinates of the respective PPI complex structure and optimized using the Protonate3D functionality from MOE [47]. Similarly, the active compounds were also docked inside their respective receptor structures. For each compound, the top scoring docked pose was retained and the electrostatic potential similarity with the ligand protein was calculated.

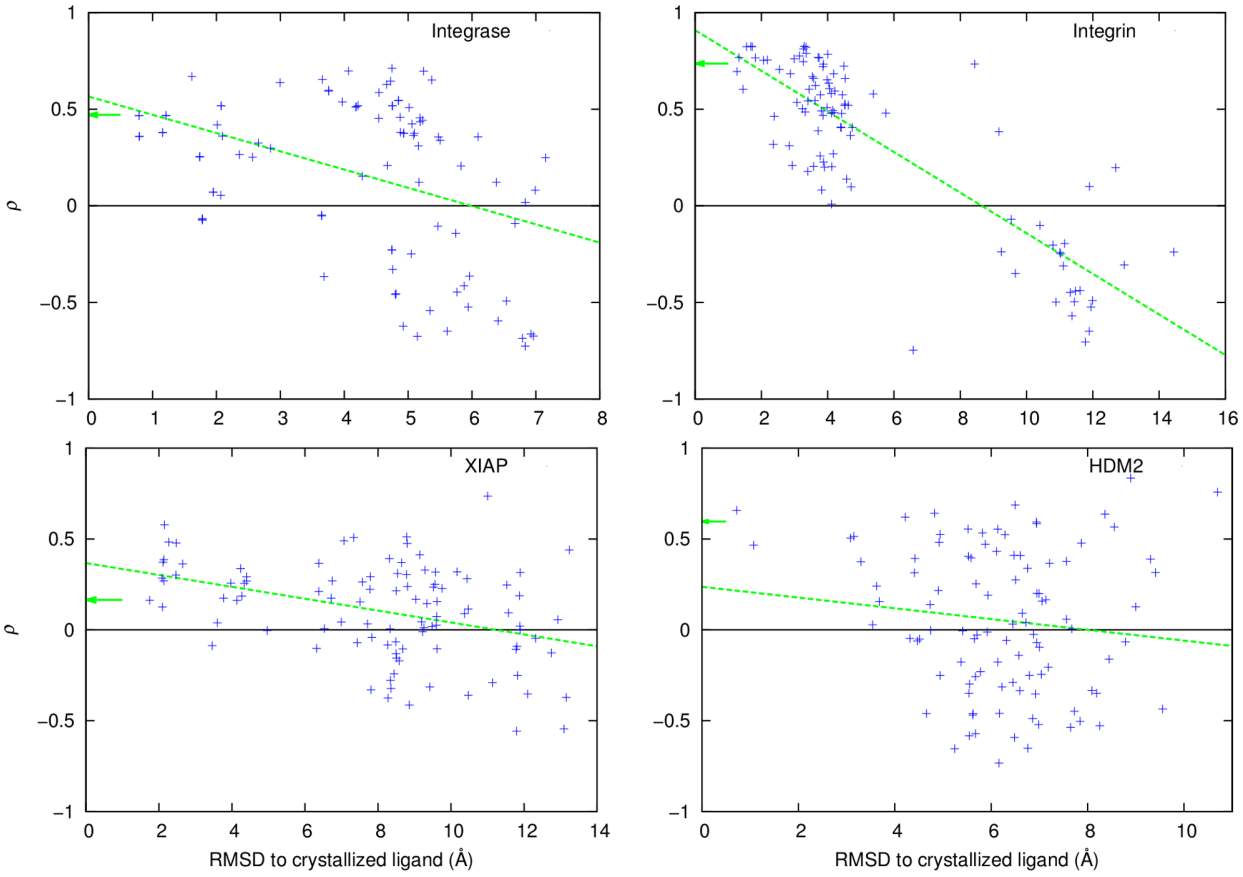


Figure 4. EleKit Spearman scores (ρ) versus RMSD to crystal structure for computationally docked poses of known active ligands. Each green arrow indicates the Spearman score for the known active ligand as seen in the crystal structure. The green dotted line was obtained using gnuplot's fit command (χ^2_{red}(Integrase)$\simeq$0.15, χ^2_{red}(Integrin)$\simeq$0.06, χ^2_{red}(XIAP)$\simeq$0.06, χ^2_{red}(HDM2)$\simeq$0.16).
doi:10.1371/journal.pone.0075762.g004

As expected, EleKit scores (based on a ligand protein and a ligand small molecule pair) are uncorrelated to docking scores (based on a ligand small molecule protein complex). The Spearman correlation between docking scores and EleKit scores fall in the range $[-0.1;0.1]$ in every case (data not shown). In all four cases, the Spearman rank correlation coefficient for the decoys follows a Gaussian distribution (figure 3). The mean of the distribution is situated at $\rho \approx 0.0$ with an exception for HIV-1 Integrase:LEDGF/p75 ($\rho \approx 0.17$). In this case, the active SMPPIIs are all mainly hydrophobic in nature except for one acid functional group, thus all decoys also bear a similar group. During the placement of those decoys into the hydrophobic pocket, this functional group has a high likelihood of adopting a similar arrangement to the active compounds, creating a higher correlating tendency and shifting the mean ρ. This distribution is independent from the scoring function as revealed by the low R^2 values for the correlation between London dG and ρ. The highest observed value was $R^2 \approx 0.006$ for the decoys of the Integrin:Fibrinogen interaction.

When SMPPIIs in their experimentally determined binding modes and with their best docked conformations are considered, a positive ρ is always observed (figure 3). Only in the case of HDM2:p53, one of the nine active molecules has a negative ρ. Furthermore, in the cases of the Integrin:Fibrinogen and Integrase:LEDGF/p75, their ρ values reside on the even higher end of the distribution. In general, a cut-off value of at least $\rho = 0$ can be suggested as it would rightfully discard approximately half of the decoys. Such filtering using ρ would only remove one inhibitor of the HDM2:p53 interaction, based on experimentally determined poses. However, the electrostatic similarity for docking results is influenced by the correctness of the predicted binding mode. In the Integrase case, the docking algorithm was unable to identify the correct binding mode within it's top solution for two compounds (the two orange arrows nearest to $\rho = 0$ in the Integrase plot of figure 3). For one of these compounds, this can be explained by the small size of the molecule (PDB: 3lpt) and its lack of an important hydrophobic group, which leaves a huge hydrophobic cavity in the protein receptor unoccupied. However, the docking algorithm forced this small inhibitor to fill the unoccupied hydrophobic cavity resulting in a wrong predicted binding mode for this inhibitor. The second compound (PDB: 4e1n) has a significantly larger substituent group and would require a minor induced fit to bind correctly. The conformational difference of the receptor protein between its ligand protein and ligand small molecule bound forms can be problematic. In the case of the XIAP:smac inhibitors, this conformational difference exists since the structure of the PPI complex was determined using NMR spectroscopy and the structures of the SMPII complexes were determined by X-ray crystallography. The hydrophobic nature of the receptor protein can be a challenge. In the HDM2:p53 interaction, only a limited number of polar interactions that might

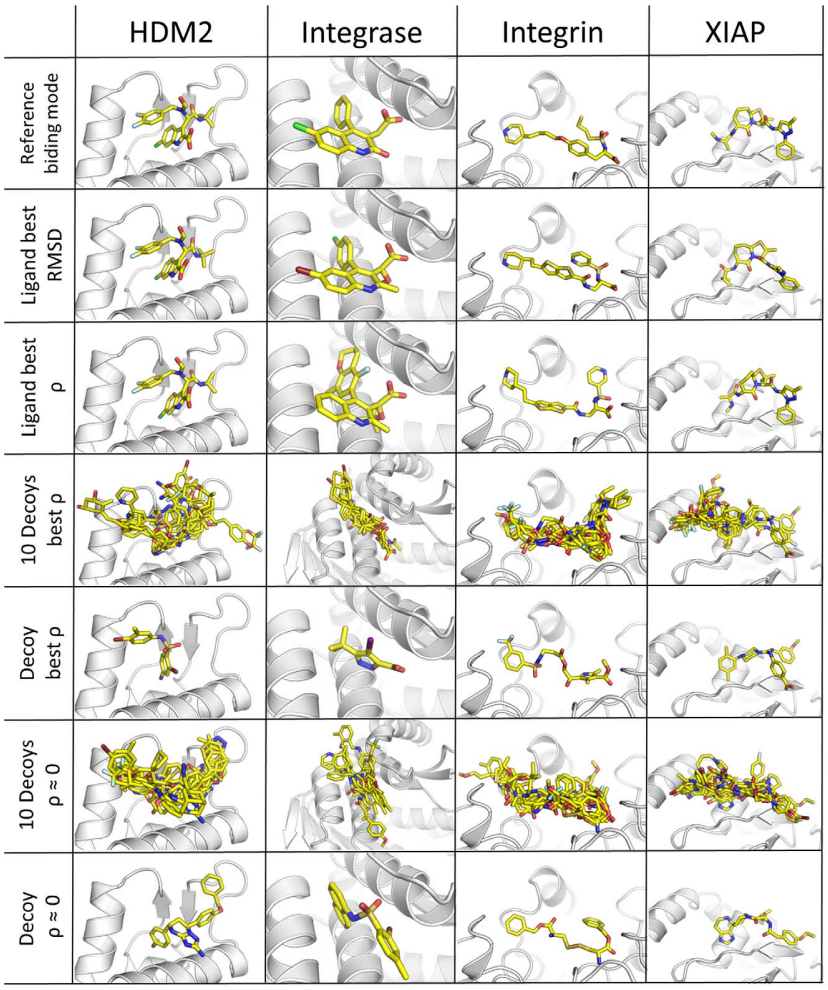


Figure 5. Analysis of docking poses. Comparison of binding modes predicted by docking for known active ligands and decoys to the reference ligands binding mode as experimentally determined. The best poses based on RMSD to the crystal structure, the pose with best ρ (ligand and decoys) and poses with ρ near zero (decoys) are shown. These binding modes indicate that the binding mode of the docked compounds are similar to the binding observed in X-ray or NMR structures. Despite the decoy compounds have binding modes in which their (a)polar contacts are similar to those of the active ligands, their electrostatic similarity with the ligand protein is different.
doi:10.1371/journal.pone.0075762.g005

help orienting the molecules in the right binding mode are present in the pocket.

An overall analysis of the docked conformation revealed that in every case the docking algorithm was able to reproduce binding modes of the active compounds in agreement with the crystallographically determined binding modes. In the four receptors examined in details (Integrase, Integrin, XIAP, HDM2), computational docking was able to place the active ligands in binding modes almost identical to those determined crystallographically (RMSD less than 1 or 2 Å, figure 4). The higher ρ even corresponds to binding modes that are closer in RMSD (Fconv [49] was used to compute RMSD for small molecules) to the experimentally determined poses. Similarly, the decoy compounds were docked within the right binding pocket making similar contacts with the receptor protein as the active compounds (figure 5), therefore validating the suitability of the docking simulations. Despite the decoy compounds made similar contacts compared to the binding modes of the active ligands, it is clear that the electrostatic similarity of the decoy compounds with the ligand protein has a normal distribution, with its mean ρ around 0. The ligands presented similar chemical groups in similar places driven by the complementarity of polar interactions in the pocket in a majority of the cases. The sole exception is found in the case of the HDM2/p53 that is hallmarked by a mainly apolar interface. Nevertheless, the apolar functions of the decoys and active ligands overlap in the binding mode.

The further away from the crystallographic pose the docked ligand is, the lower the Spearman rank correlation becomes. As a remark, the Receiver Operating Characteristic (ROC) analysis is typically used to assess the predictive and enrichment power of a method. But due to the lack of a significant number of active SMPPIIs for which structural information is available for a single target, this type of analysis could not be performed.

The development of EleKit was inspired by the computational work on electrostatic complementarity at protein-protein interfaces by McCoy *et al.* [17]. But EleKit bears salient differences with this former study. Whereas McCoy *et al.* studied the complementarity of protein-protein interfaces, EleKit measures the local similarity between one ligand protein and small molecules targeting the same receptor interface. McCoy *et al.* measured the

correlation of electrostatic potentials at molecular surface points while EleKit works on a 3D volume in the solvent region near the binding interface. There are some significant prior works that compare electrostatic potentials and other molecular interaction fields for proteins only [18–21,37,50].

EleKit was also inspired by the commercial tools EON [27–29] and ElectroShape [30]. In effect, these tools can assess similarity of electrostatic potentials. However, they work exclusively on small molecules. Thus, they can identify compounds similar to a known small molecule inhibitor, but this is not applicable when searching for a first-in-class SMPPII [31].

Conclusions

We have developed a method (EleKit) to investigate the similarity between protein ligands and their respective SMPPIIs. Our analysis of available SMPPII structures indicates that in cases where SMPPIIs bind without induced fit, there is similarity of electrostatic potentials at the interacting interface between a protein ligand and a small molecule inhibitor. This insight can be applied to post-filter virtual screening results to remove unpromising compounds and thus has some potential for the rational design of novel SMPPIIs.

Acknowledgments

We thank RIKEN, Japan, for an allocation of computing resources on the RIKEN Integrated Cluster of Clusters (RICC). Dr. Eleanor Banwell is acknowledged for proofreading the manuscript.

Author Contributions

Conceived and designed the experiments: AV FB KYJZ. Performed the experiments: AV FB. Analyzed the data: AV FB KYJZ. Wrote the paper: AV FB KYJZ.

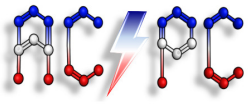
References

1. Stumpf MPH, Thorne T, de Silva E, Stewart R, An HJ, et al. (2008) Estimating the size of the human interactome. Proc Natl Acad Sci U S A 105: 6959–64.
2. Venkatesan K, Rual JF, Vazquez A, Stelzl U, Lemmens I, et al. (2009) An empirical framework for binary interactome mapping. Nat Meth 6: 83 – 90.
3. Wells JA, McClendon CL (2007) Reaching for high-hanging fruit in drug discovery at protein-protein interfaces. Nature 450: 1001–1009.
4. Verdine GL, Walensky LD (2007) The Challenge of Drugging Undruggable Targets in Cancer: Lessons Learned from Targeting BCL-2 Family Members. Clin Cancer Res 13: 7264–7270.
5. Villoutreix BO, Labbe CM, Lagorce D, Laconde G, Sperandio O (2012) A Leap into the Chemical Space of Protein-Protein Interaction Inhibitors. Curr Pharm Des 18: 4648–67.
6. Morelli X, Bourgeas R, Roche P (2011) Chemical and structural lessons from recent successes in protein-protein interaction inhibition (2P2I). Curr Opin Chem Biol 15: 475 – 81.
7. Dias R, de Azevedo WFJ (2008) Molecular Docking Algorithms. Curr Drug Targets 9: 1040–7.
8. Horvath D (2011) Pharmacophore-Based Virtual Screening, volume 672 of *Methods Mol Biol*. Humana Press, 261–98 pp.
9. Fry DC (2008) Drug-like inhibitors of protein-protein interactions: A structural examination of effective protein mimicry. Curr Protein Pept Sci 9: 240–7.
10. Fry DC (2012) Small-Molecule Inhibitors of Protein-Protein Interactions: How to Mimic a Protein Partner. Curr Pharm Des 18: 4679–84.
11. Voet A, Zhang KYJ (2012) Pharmacophore Modelling as a Virtual Screening Tool for the Discovery of Small Molecule Protein-protein Interaction Inhibitors. Curr Pharm Des 18: 4586–98.
12. Voet A, Banwell E, Sahu K, Heddle J, Zhang KYJ (2013) Protein interface pharmacophore mapping tools for small molecule protein: Protein interaction inhibitor discovery. Curr Top Med Chem 13: 989–1001.
13. Nray-Szab G (1993) Analysis of molecular recognition: steric electrostatic and hydrophobic complementarity. J Mol Recognit 6: 205–10.
14. Nicholls A, Honig B (1991) A rapid finite difference algorithm, utilizing successive over-relaxation to solve the PoissonBoltzmann equation. Journal of Computational Chemistry 12: 435–445.
15. Baker NA, Sept D, Joseph S, Holst MJ, McCammon JA (2001) Electrostatics of nanosystems: application to microtubules and the ribosome. Proc Natl Acad Sci U S A 98: 10037–41.
16. Rohs R, West SM, Sosinsky A, Liu P, Mann RS, et al. (2009) The role of DNA shape in protein-DNA recognition. Nature 461: 1248–1253.
17. McCoy AJ, Chandana Epa V, Colman PM (1997) Electrostatic complementarity at protein/protein interfaces. J Mol Biol 268: 570–84.
18. Blomberg N, Gabdoulline RR, Nilges M, Wade RC (1999) Classification of protein sequences by homology modeling and quantitative analysis of electrostatic similarity. Proteins: Structure, Function, and Bioinformatics 37: 379–387.
19. Dlugosz M, Trylska J (2008) Electrostatic similarity of proteins: Application of three dimensional spherical harmonic decomposition. The Journal of Chemical Physics 129: 015103.
20. Kieslich CA, Morikis D, Yang J, Gunopulos D (2011) Automated computational framework for the analysis of electrostatic similarities of proteins. Biotechnology Progress 27: 316–325.
21. Gorham R, Kieslich C, Morikis D (2011) Electrostatic Clustering and Free Energy Calculations Provide a Foundation for Protein Design and Optimization. Annals of Biomedical Engineering 39: 1252–1263.
22. Janin J, Chothia C (1990) The structure of protein-protein recognition sites. Biol Chem 265: 16027–30.
23. Jones S, Thornton JM (1995) Protein-protein interactions – a review of protein dimer structures. Progr Biophy molec Biol 63: 31–65.
24. Chau PL, Dean PM (1994) Electrostatic complementarity between proteins and ligands. 1. Charge disposition, dielectric and interface effects. J Comput-Aided Mol Des 8: 51325.
25. Kieslich CA, Morikis D (2012) The Two Sides of Complement C3d: Evolution of Electrostatics in a Link between Innate and Adaptive Immunity. PLoS Comput Biol 8: e1002840.
26. Dlugosz M, Antosiewicz JM, Zielinski P, Trylska J (2012) Contributions of Far-Field Hydrodynamic Interactions to the Kinetics of Electrostatically Driven Molecular Association. J Phys Chem B 116: 5437–5447.
27. Muchmore SW, Souers AJ, Akritopoulou-Zanze I (2006) The Use of Three-Dimensional Shape and Electrostatic Similarity Searching in the Identification of a Melanin-Concentrating Hormone Receptor 1 Antagonist. Chem Biol Drug Des 67: 174–176.
28. Naylor E, Arredouani A, Vasudevan SR, Lewis AM, Parkesh R, et al. (2009) Identification of a chemical probe for NAADP by virtual screening. Nat Chem Biol 5: 220–226.
29. OpenEye Scientific Software, Inc (2010) OEChem 1.7.4. Santa Fe, NM, USA: OpenEye Scientific Software, Inc.
30. Armstrong M, Morris G, Finn P, Sharma R, Moretti L, et al. (2010) ElectroShape: fast molecular similarity calculations incorporating shape, chirality and electrostatics. Journal of Computer-Aided Molecular Design 24: 789–801.
31. Cavalluzzo C, Voet A, Christ F, Singh BK, Sharma A, et al. (2012) De novo design of small molecule inhibitors targeting the LEDGF/p75-HIV integrase interaction. RSC Advances 2: 974–84.
32. Dolinsky TJ, Nielsen JE, McCammon JA, Baker NA (2004) PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. Nucleic Acids Res 32: W665–7.
33. Carbó R, Leyda L, Arnau M (1980) How similar is a molecule to another? An electron density measure of similarity between two molecular structures. International Journal of Quantum Chemistry 17: 1185–1189.
34. Hodgkin EE, Richards WG (1987) Molecular similarity based on electrostatic potential and electric field. International Journal of Quantum Chemistry 32: 105–110.
35. Leroy X, Doligez D, Frisch A, Garrigue J, Rémy D, et al. (2011) The OCaml system release 3.12 Documentation and user's manual. INRIA, France.
36. Danelutto M, Di Cosmo R (2012) A "Minimal Disruption" Skeleton Experiment: Seamless Map and Reduce Embedding in OCaml. Procedia Computer Science 9: 1837 – 1846.
37. Wade RC, Gabdoulline RR, De Rienzo F (2001) Protein interaction property similarity analysis. International Journal of Quantum Chemistry 83: 122–127.
38. Davis ME, Madura JD, Luty BA, McCammon J (1991) Electrostatics and diffusion of molecules in solution: simulations with the University of Houston Brownian dynamics program. Computer Physics Communications 62: 187 – 197.
39. Petke J (1993) Cumulative and discrete similarity analysis of electrostatic potentials and fields. Journal of Computational Chemistry 14: 928–933.
40. Hartman G, Egbertson M, Halczenko W, Laswell WL, Duggan ME, et al. (1992) Non-peptide fibrinogen receptor antagonists. 1. Discovery and design of exosite inhibitors. J Med Chem 35: 46402.
41. Christ F, Voet A, Marchand A, Nicolet S, Desimmie B, et al. (2010) Rational design of small-molecule inhibitors of the LEDGF/p75-integrase interaction and HIV replication. Nat Chem Biol 6: 442–8.
42. Arkin MR, Randal M, DeLano WL, Hyde J, Luong TN, et al. (2003) Binding of small molecules to an adaptive proteinprotein interface. Proc Natl Acad Sci U S A 100: 1603–8.
43. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical Recipes 3rd Edition: The Art of Scientific Computing. New York, NY, USA: Cambridge University Press.

44. Irwin JJ, Shoichet BK (2005) ZINC A Free Database of Commercially Available Compounds for Virtual Screening. J Chem Inf Model 45: 177–82.
45. Huang N, Shoichet BK, Irwin JJ (2006) Benchmarking Sets for Molecular Docking. J Med Chem 49: 6789–801.
46. Mysinger MM, Carchia M, Irwin JJ, Shoichet BK (2012) Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking. J Med Chem 55: 6582–94.
47. Chemical Computing Group Inc (2011) Molecular Operating Environment (MOE) 10th edition. 1010 Sherbooke St. West, Suite #910, Montreal, QC, Canada, H3A 2R7: Chemical Computing Group Inc.
48. Naim M, Bhat S, Rankin KN, Dennis S, Chowdhury SF, et al. (2007) Solvated interaction energy (SIE) for scoring protein-ligand binding affinities. 1. Exploring the parameter space. J Chem Inf Model 47: 122–33.
49. Neudert G, Klebe G (2011) fconv: Format conversion, manipulation and feature computation of molecular data. Bioinformatics 27: 1021–2.
50. Ullmann GM, Hauswald M, Jensen A, Kostić NM, Knapp EW (1997) Comparison of the Physiologically Equivalent Proteins Cytochrome c6 and Plastocyanin on the Basis of Their Electrostatic Potentials. Tryptophan 63 in Cytochrome c6 May Be Isofunctional with Tyrosine 83 in Plastocyanin. Biochemistry 36: 16187–16196.

3.4 Descripteur moléculaire pour le criblage virtuel à haute vitesse

> « C'est ce qui est invariant qui est important ».
>
> Hai-quan Yang



Des mesures de similarité ont été développées depuis les tout débuts de la chemoinformatique. Ces mesures peuvent porter sur des descripteurs moléculaires, des fragments de molécules, des similarités de graphes ou de surfaces.

Les mesures de similarité sont largement utilisées en pratique et ont démontré leur utilité dans la découverte de médicaments.

De par notre intérêt pour la mesure de similarité dans l'espace électrostatique, ainsi que pour le criblage virtuel à haute vitesse, nous avons cherché à exploiter au maximum l'information contenue dans les coordonnées des atomes d'une molécule ainsi que leurs charges partielles.

Nous proposons ici un descripteur moléculaire basé sur les charges partielles et la configuration tridimensionnelle d'une molécule. Notre descripteur utilise la fonction d'autocorrélation et une fonction triangulaire noyau afin d'encoder tous les atomes d'une molécule en deux vecteurs invariants à la translation et à la rotation de la molécule encodée (figure 1.4).

Combiné à une fonction de score, ce descripteur permet d'ordonner une base de données de molécules en fonction d'une molécule requête. Le résultat attendu étant que d'autres molécules actives se trouveront en tête de la liste ainsi obtenue. Le logiciel que nous proposons s'appelle ACPC (« Auto-Correlation of Partial Charges », soit en français « Auto-Corrélation des Charges Partielles »).

De nombreuses expériences de criblage virtuel *in-silico* à posteriori ont été menées et d'autres méthodes ont été comparées afin de valider notre approche ainsi que son protocole d'utilisation recommandé.

Alors que la méthode est simple dans son principe, elle se comporte remarquablement lors des expériences. À une vitesse moyenne de 1649 molécules comparées par seconde (encodage inclus), la méthode atteint une aire sous la courbe ROC médiane de 0.81 sur un jeu de 40 protéines cibles différentes.

Berenger *et al. Journal of Cheminformatics* 2014, **6**:23
http://www.jcheminf.com/content/6/1/23

Journal of Cheminformatics

RESEARCH ARTICLE Open Access

A rotation-translation invariant molecular descriptor of partial charges and its use in ligand-based virtual screening

Francois Berenger, Arnout Voet, Xiao Yin Lee and Kam YJ Zhang*

Abstract

Background: Measures of similarity for chemical molecules have been developed since the dawn of chemoinformatics. Molecular similarity has been measured by a variety of methods including molecular descriptor based similarity, common molecular fragments, graph matching and 3D methods such as shape matching. Similarity measures are widespread in practice and have proven to be useful in drug discovery. Because of our interest in electrostatics and high throughput ligand-based virtual screening, we sought to exploit the information contained in atomic coordinates and partial charges of a molecule.

Results: A new molecular descriptor based on partial charges is proposed. It uses the autocorrelation function and linear binning to encode all atoms of a molecule into two rotation-translation invariant vectors. Combined with a scoring function, the descriptor allows to rank-order a database of compounds versus a query molecule. The proposed implementation is called ACPC (AutoCorrelation of Partial Charges) and released in open source. Extensive retrospective ligand-based virtual screening experiments were performed and other methods were compared with in order to validate the method and associated protocol.

Conclusions: While it is a simple method, it performed remarkably well in experiments. At an average speed of 1649 molecules per second, it reached an average median area under the curve of 0.81 on 40 different targets; hence validating the proposed protocol and implementation.

Keywords: RTI molecular descriptor; Partial charges; Ligand-based virtual screening; Spatial auto-correlation; Cross-correlation; Linear binning; ACPC

Background

Molecular similarity is a widely studied topic in chemoinformatics [1-3] and medicinal chemistry [4]. Molecular similarity is used in ligand-based virtual screening [5-8], SAR by catalog and to predict side effects. When hits are obtained in a drug discovery project, it is interesting to test similar derivatives in the hope that some will be more potent. Various molecular similarity measures have been developed [9]. Those include measures based on molecular descriptors [10], measures based on molecular fragments (such as MACCS) and measures based on graph matching (such as maximum common substructure searches [11-13]). For an extensive reference on molecular descriptors, cf. [10]. The DRAGON software [14] can compute thousands of such descriptors.

Electrostatics are one of the main driving forces of molecular recognition, along with steric complementarity, hydrogen bonding and hydrophobic interactions [15]. It is well known that there is complementarity in shape and electrostatics between a ligand molecule and its receptor protein. Molecules sharing similar electrostatics and shape are expected to bind to the same receptor. This principle has been used in ligand-based virtual screening to look for small molecules similar to known inhibitors and natural substrates [16-19]. The electrostatic potential of a molecule originates from the atomic partial charges. While the electrostatic reaches into the long range, the

*Correspondence: kamzhang@riken.jp
Zhang Initiative Research Unit, Institute Laboratories, RIKEN, 2-1 Hirosawa, Wako, Saitama 351-0198, Japan

© 2014 Berenger et al.; licensee Chemistry Central Ltd. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated.

Chemistry Central

partial charges also contribute to the molecular recognition in the short range where they are the driving forces of polar interactions (hydrogen bonding and salt bridges) determining the specificity of recognition as well as the coordination of bridging waters between the receptor and ligand. Due to our recent study on the electrostatic similarity of small molecules with binding proteins [20], we were interested in the challenge of looking for active compounds while only exploiting the information contained in atomic coordinates and partial charges of a molecule.

The autocorrelation function has long been used in chemoinformatics, by several researchers and in various ways [21-27]. In 1980, Gilles Moreau and Pierre Broto proposed to use the autocorrelation function on the molecular graph to encode any property associated with atoms [21]. They subsequently used their autocorrelation descriptor in SAR studies [28,29]. The atom-type autocorrelation (ATAC) descriptors sum property values of atoms with given types [10]. Atom pairs [22] and Chemically Advanced Template Search (CATS) [30] are examples of such descriptors. The CATS2D descriptor [30], later extended to 3D [26], computes the topological cross-correlation of generalized atom types (hydrogen bond donor/acceptor, positively/negatively charged, lipophilic). The autocorrelation of partial charges of a 3D molecule encoded into histograms was studied [25]. Histograms of the autocorrelation of molecular surface properties were also used in partial least squares analysis to generate ligand-based 3D-QSARs [27], in PCA analysis [23], in Kohonen maps and in training neural networks for the prediction of biological activity [23,24].

A new molecular descriptor based on partial charges is proposed. It uses the autocorrelation function to encode all atoms of a molecule into two rotation-translation invariant vectors. The method is named ACPC (Auto-Correlation of Partial Charges). Compared to previous methods, neither using histograms nor explicitly computing the electrostatic potential field, but splitting the autocorrelation values based on their sign before applying linear binning [31] with a thin discretization step are essential traits of ACPC. Combined with a scoring function, it can rank-order a database of compounds versus a query molecule. The descriptor is solely based on the 3D distribution of partial charges, uses a single discretization parameter and provides good performance and speed. It was tested in a retrospective ligand-based virtual screening setting. At an average speed of 1649 molecules per second, it reached an average median Area Under the ROC[a] Curve (AUC) of 0.81 on 40 different targets, outperforming several commonly used methods and making it a useful addition to the arsenal of ligand based virtual screening tools.

Method

The point charge model of electrostatics is considered. In this model, each atom is a point in 3D space with an associated partial charge. The ACPC method first computes the autocorrelation of partial charges of all atoms in a molecule. Then, it separates the positive from negative values before applying linear binning. To score molecules, the sum of cross-correlations at lag zero is computed for corresponding vectors of the two molecules under consideration. A mathematical description follows and a graphical overview can be seen in Figure 1.

Algorithm

Let m be a molecule with A atoms.
Let $i = (x_i, y_i, z_i, q_i)$ be the atom at index i in m ($1 <= i <= A$), with 3D coordinates (x_i, y_i, z_i) and partial charge q_i.
Let d_{ij} be the Euclidean distance between atoms i and j of m.
Let k be the lag (an inter-atomic distance in fact).
Let $\delta_{kd_{ij}}$ be the Kronecker delta equal to one if $k = d_{ij}$, zero otherwise.

Therefore, the autocorrelation of molecule m can be written as

$$AC(m,k) = \sum_{i=1}^{A} \sum_{j=1}^{A} q_i q_j \delta_{kd_{ij}}$$

Since in practice values of the autocorrelation vector for $k = 0$ are ignored and each atom pair is considered only once $(i, j = j, i)$, the following formula is used:

$$ACPC(m,k) = \sum_{i=1}^{A-1} \sum_{j=i+1}^{A} q_i q_j \delta_{kd_{ij}}$$

Since the autocorrelation values can be positive or negative, they are split based on their sign before discretization, in order to avoid cancellation of values. This gives the two vectors AC^+ and AC^-.

$$(AC^+(m,k), AC^-(m,k)) = sign\_split(ACPC(m,k))$$

such that $AC^+(m,k)$ is the subset of $ACPC(m,k)$ where $q_i * q_j >= 0$ and $AC^-(m,k)$ is the subset where $q_i * q_j < 0$.

$$ACPC(m,k) = AC^+(m,k) \cup AC^-(m,k)$$

To convert these two sets into vectors, linear binning [31] is used. Linear binning consists in selecting a discretization step (dx) for an axis, then linearly interpolating each raw data point's contribution to the two neighboring points on the discretized axis. In our case, a raw data point is a pair of (d_{ij}, q_{ij}). Assuming that d_{ij} is between x and z (with $z = x + dx$) on the discretized axis, q_{ij} is separated

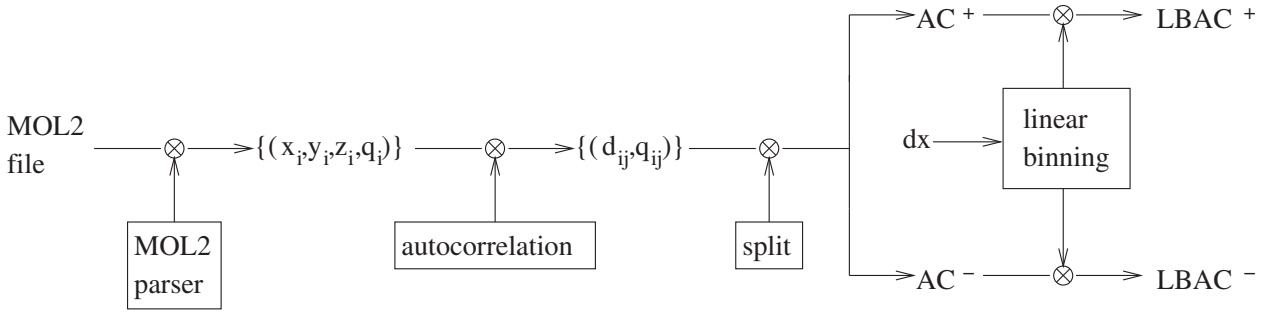


Figure 1 Overview of the ACPC method. It encodes a 3D molecule into a rotation-translation invariant molecular descriptor based on partial charges and inter-atomic distances. The descriptor consists of two vectors separating the positive from the negative autocorrelation values. These vectors are discretized by linear-binning with discretization step *dx* in order to obtain the final pair of vectors ($LBAC^+$ and $LBAC^-$).

into a contribution at x (called c_x) and a contribution at z (called c_z) by applying the formulae

$$c_x = q_{ij} * \frac{z - d_{ij}}{dx} \text{ and } c_z = q_{ij} * \frac{d_{ij} - x}{dx}.$$

The result of linear binning is a vector containing the sum of all these contributions. Therefore, by applying linear binning, two rotation-translation invariant vectors encoding a molecule are obtained:

$$LBAC^+(m,k,dx) = linbin(AC^+(m,k),dx)$$

$$LBAC^-(m,k,dx) = linbin(AC^-(m,k),dx)$$

By default, to score the similarity between two linearly-binned sign-split autocorrelation vectors, cross correlation at lag zero is used. Molecules being encoded in a rotation-translation invariant way, there is no need to scan for the lag which would yield the maximum cross-correlation. This maximum would be at lag zero for similar molecules. Since the goal is to assess the similarity of molecules, scanning the lag would be inappropriate.

Let n be another molecule.

Let $l1 = length(LBAC^+(m,k,dx))$ and $l2 = length(LBAC^+(n,k,dx))$ in $p = min(l1,l2)$ and let $l3 = length(LBAC^-(m,k,dx))$ and $l4 = length(LBAC^-(n,k,dx))$ in $q = min(l3,l4)$.

The cross-correlation at lag zero of molecule m with molecule n is $CC(m,n,dx) = \sum_{k=0}^{p} LBAC^+(m,k,dx)LBAC^+(n,k,dx) + \sum_{k=0}^{q} LBAC^-(m,k,dx)LBAC^-(n,k,dx)$.

The spectrum obtained for one molecule, before and after linear binning can be seen in Figure 2. Differences in peak heights between $AC^{\pm}$ and $LBAC^{\pm}$ are due to linear binning. It is noteworthy that such spectra are not completely uninterpretable. Each Kronecker delta could be textually annotated by the names (from the MOL2 file for example) of the two atoms that give birth to it. Of course, such annotations would be crowded near $y = 0$ but readable for peaks which are standard deviations away from the mean.

Software features

ACPC allows rank-ordering of a compound database against a query molecule. By default, ACPC filters out multiple conformers of the same molecule[b] by keeping only the best scoring one. Molecular descriptors are computed and scored on the fly. Molecule names, scores and ranks are written to disk. A special program (acpc_big) is also provided to score large databases. In contrary to the default program, acpc_big executes in constant memory space but does not rank nor filter out multiple conformers. For tests on datasets with known active compounds (whose molecule names must be prefixed with the word "active"), the AUC is computed by the CROC package [32]. Several scoring functions are available (CC, Tanimoto, $Tversky_{ref}$, $Tversky_{DB}$). The following remaining features are optional. The computation can be parallelized on multicore computers using the Parmap library [33] for multiple queries on the same database. The filtering of multiple conformers can be turned off. The ROC and

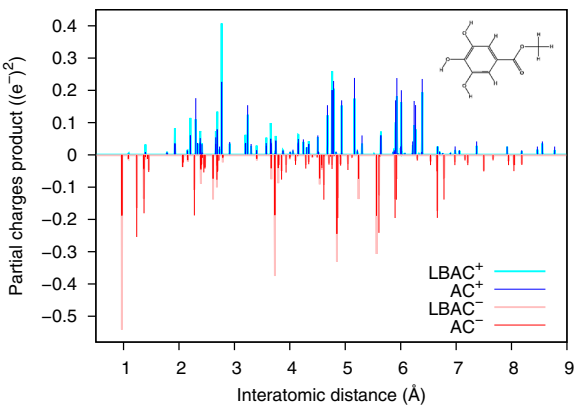


Figure 2 Autocorrelogram. In blue and red impulses: positive and negative values of the autocorrelation ($AC^{\pm}$) of partial charges for the first ligand of the comt target. The ligand is depicted in 2D at the top right. The linearly binned versions ($LBAC^{\pm}$) of the positive and negative parts of the autocorrelogram are the cyan and pink lines.

AUC calculations can be turned off. The ROC curve can be displayed on screen with Gnuplot [34]. The discretization step can be changed.

Results and discussion

Datasets

To evaluate ACPC, two different datasets were prepared. All ligands and decoys come from release two of the Directory of Useful Decoys (DUDr2) [35]. The DUD contains 2950 ligands for 40 different targets. Every ligand has 36 decoy molecules that are physically similar but topologically distinct. OMEGA v2.4.6 [36] was used to generate conformers. The first dataset (named "1conf") contains a single, lowest energy conformer per molecule. It is further filtered to have unique SMILES. If several molecules have the same SMILES string as assigned by Open Babel [37], only the first encountered molecule was kept. This allows to easily detect and filter out multiple conformers of the same molecule later on, when one wants to keep only the highest scoring conformer. Finally, MOE [38] in combination with its default force-field (MMFF94x) was used to assign partial charges to molecules. As "1conf" is of reasonable size for each target, it is affordable to test each active of a given target in turn as a query, especially for the four software that run well in a cluster environment (ACPC, Pharao, Open Babel and Shape-it). The second dataset (named "25conf") is the expansion of "1conf" by generating a maximum of 25 low energy conformers per molecule. As "25conf" is a much bigger dataset, each software is run only once with a single query per target. On "25conf", it is feasible to test all the MOE fingerprints since a single query per target is performed. Cf. section "Availability and requirements" to download these datasets.

Parametrization

The discretization parameter (dx) was setup using 20 randomly selected targets (from a total of 40). Then, for each selected target, only the first ligand in its ligands list was used to measure the AUC reached by ACPC. The dx parameter is introduced because of linear binning [31]. dx allows to balance the trade-off between the speed of the algorithm and the approximation error introduced by binning [31]. Small values of dx are important in our method since they counterbalance the information loss incurred by dimensionality reduction.

The default value proposed ($dx = 0.005$Å) gives a good compromise between speed and average AUC reached. dx values in the range $0.001 <= dx <= 0.009$ are acceptable, but 0.001 consumes too much memory and values over 0.009 don't perform as well in terms of AUC (Table 1).

Concerning the charge model used to assign partial charges; the default charge model from MOE (MMFF94x) was used in all experiments. For users with no access to MOE, the Gasteiger charge model [39] from Open Babel performs nearly as well. Other charge models available in Open Babel (QTPIE [40], QEq [41], MMFF94 [42,43]) perform worse and hence are not recommended for use with ACPC (cf. Table 2).

Table 1 Effect of the *dx* parameter

dx (Å)	Average (AUCs)	Median (AUCs)
0.005	0.73	0.75
0.01	0.69	0.71
0.05	0.68	0.70
0.1	0.67	0.70
0.5	0.66	0.66

Test protocol: 20 targets and five queries per target were randomly chosen on "1conf".

Additional experiments were carried out retrospectively to confirm the choices of parameters. The test protocol was as follows: 20 targets and five queries per target were randomly selected on "1conf". Then, the effect of the sign_split function, the dx parameter and the force field used to assign partial charges were measured in three distinct experiments. Those results are shown in Tables 1, 2 and 3. As can be seen from Table 3, the use of the sign_split function gave better AUCs. The finer discretization parameter gave the best AUCs compared to coarser ones (Table 1). Using the MMFF94x force field to assign partial charges gave the best AUCs compared to other force fields (Table 2).

Validation protocol

ACPC was compared against a diverse set of freely available methods and to the molecular fingerprints available in MOE.

The open source software compared against are: 1) the Pharmacophore alignment and optimization tool Pharao [44], from Silicos-it [45] 2) the purely shape-based tool Shape-it, also from Silicos-it [46] and 3) the MACCS fingerprint as implemented in Open Babel [37].

Pharao is an open source software to align and score small molecules using pharmacophores. Pharao uses 3D

Table 2 Effect of the force field used to assign partial charges (OB stands for Open Babel)

Force field	Average (AUCs)	Median (AUCs)
MOE's MMFF94x	0.77	0.84
OB's Gasteiger	0.75	0.77
OB's MMFF94	0.72	0.75
OB's QEQ	0.70	0.72
OB's QTPIE	0.69	0.74

Test protocol: 20 targets and five queries per target were randomly chosen on "1conf".

Table 3 Effect of the sign_split function

	With	Without
Average (AUCs)	0.77	0.76
Median (AUCs)	0.83	0.75

Test protocol: 20 targets and five queries per target were randomly chosen on "1conf".

Gaussians to represent pharmacophore features instead of the more common points and spheres model. Pharao's performance has been demonstrated in virtual screening experiments and unsupervised clustering of small molecules [44]. Tversky_ref is recommended with Pharao to score compounds in virtual screening experiments [44].

Shape-it is a tool to align a reference molecule against a database of molecules. Shape-it uses 3D Gaussians to describe the molecular shape [47]. Shape-it can find the alignment of molecules which maximizes their volume overlap. Tversky_ref was used to score compounds with Shape-it, since it gave better results than Tanimoto or Tversky_db.

The MACCS fingerprint is a bit string registering the presence or absence of structural features (MACCS stands for Molecular ACCess System, originally developed by Molecular Design Limited, now Accelrys). In Open Babel, Tanimoto is used to score compounds with MACCS.

All ligand fingerprints available in MOE v2013.08 [38] have also been used. Some of the MOE fingerprints (TAD, TAT, TGT, TGD) can be traced back to the literature [48,49]. All MOE fingerprints use Tanimoto to score compounds, except ESshape3D and ESshape3d_HYD which use an inverse distance. If an abbreviated name is used in tables, it is given between parentheses below.

- ESshape3D_HYD (ES3DH) is an eigenvalue spectrum shape fingerprint. It allows for comparison of 3D shapes made by hydrophobic heavy atoms of a molecule.
- ESshape3D (ES3D) is similar to ESshape3D_HYD but uses all heavy atoms instead of just hydrophobic ones.
- GpiDAPH3 (Gpi3) is a three points pharmacophore fingerprint calculated from the 2D molecular graph. Each atom is given one of eight atom types computed from three atomic properties (in pi system, is donor, is acceptor). Anions and cations are ignored.
- piDAPH3 (pi3) is similar to GpiDAPH3 but uses the molecule's 3D conformation instead of the 2D molecular graph.
- piDAPH4 (pi4) is similar to piDAPH3 but considers quadruplets of pharmacophore features instead of triangles.
- TAD is a two points pharmacophore fingerprint calculated from the molecule's 3D conformation. It considers pairs of pharmacophore features (donor, acceptor, polar, anion, cation, hydrophobic).
- TAT is similar to TAD but uses triangles instead of point pairs.
- TGD is similar to TAD but uses the 2D molecular graph instead of the 3D conformation.
- TGT is similar to TGD but uses triangles instead of point pairs.

PAR [50] was used to accelerate some experiments by parallelizing their execution on a multicore computer.

Performance

As a reminder about reading AUC values from ROC curves: $AUC = 0.5$ is the performance of a random method. An AUC score of less than 0.5 means that a method perform worse than random. All rationally engineered methods are expected to perform significantly above 0.5 in terms of AUC.

Test results using all possible queries on the "1conf" dataset are shown in Figure 3 as a per target quartile plot and in Table 4 as median AUCs, while Figure 4 gives an aggregated overview.

In Figure 3, three classes are distinguishable. Class 1: ACPC is far in front. Class 2: ACPC's performance ties or significantly overlaps with at least one other method. Class 3: ACPC is outperformed by at least one other method. Class 1 targets (17 cases): gart, dhfr, parp, pnp, sahn, tk, gbp, fxa, cox2, fgfr1, src, cdk2, hivrt, thrombin, pr, vegfr2 and ache. Class 2 targets (9 cases): tie in eragonist, na, egfr, ppargamma, trypsin, ampc, ada, ar and pde5. Class 3 targets (13 cases): hmga, rxralpha, comt, mr, gr, hivpr, hsp90, alr2, inha, erantagonist, p38, pdgfrb, ace. Cox1 is classified as an exception as no method perform well on this target: ACPC, Pharao and Shape-it all have median AUCs below 0.5 and MACCS has a huge standard deviation with a mean not far from 0.5. By looking at the cumulative distribution function (CDF) from experiments on "1conf" in Figure 4, such statements can directly be read: the probability for ACPC to have an AUC less than 0.5 is about 10%. All other software have a higher probability to have an AUC less than 0.5. By looking at the CDF, the previous statement is seen to hold for ACPC for any $AUC >= 0.41$. The best performance is reached 20 times by ACPC, 13 times by MACCS, seven times by Pharao and three times by Shape-it.

Test results on the "25conf" dataset are shown in Table 5. In terms of average AUC reached, the best method is ACPC (0.78) followed by GpiDAPH3 (0.72) then MACCS (0.7). The best AUC is reached 17 times by ACPC, five times by Pharao and four times by the MOE fingerprints TAD and TAT. On average, the GpiDAPH3 fingerprint from MOE is the second best method on this dataset. In order to confirm the stability of the test results on the "25conf" dataset using a single query, ACPC was run with all the active ligands as queries. The resulting

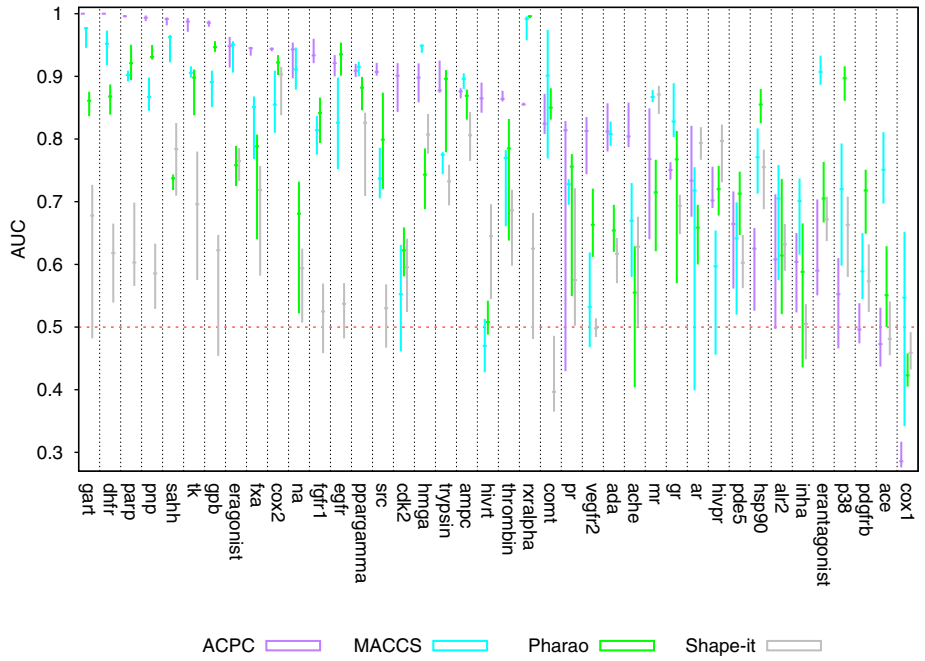


Figure 3 1st, 2nd and 3rd quartile plots for ACPC, MACCS, Pharao and Shape-it on all queries of the "1conf" dataset. For clarity, targets are sorted on the x axis by decreasing median value obtained by ACPC. The red dotted horizontal line at AUC = 0.5 indicates random performance.

average AUCs are shown in column $ACPC_{25c}$ in Table 5. For comparison, the average AUCs from the "1conf" dataset are also calculated and shown in columns $ACPC_{1c}$ in Table 5. The performance of ACPC on "25conf" using a single query is stable, since the average AUCs are comparable to that using all the actives as queries. Moreover, the performance of ACPC using either a single query or all actives as queries is similar to that from the "1conf" dataset.

Speeds of the software ran on "1conf" are shown in Figure 5. Speed tests were performed on one core of a 2.4 GHz Intel Xeon workstation with 12 GB of RAM running Ubuntu Linux 12.04. Speed measurements were done on the egfr target, since it has the most ligands and decoys. Reported numbers were averaged over three runs. Only scoring was performed by each method, no ranking or filtering of compounds was done. ACPC and Shape-it screen a database of compounds read from a MOL2 file. Open Babel reads compounds from a SMILES file. Pharao reads compounds from a .phar file (its own text format for pharmacophore features). ACPC processed 1649 molecules per second, Pharao 1416, Open Babel 553 and Shape-it 65.

The scaffold diversity among actives in the top 10 to 50 ranked molecules was also investigated for cox2 and egfr, the two targets with the most ligand clusters (44 and 40 respectively). Andrew Good's clustering analysis of DUD (http://dud.docking.org/clusters/) was used to assign each active to a cluster via reduced graphs [51]. Ten random queries were used on each target then cluster of actives in the top 10 to 50 molecules were analyzed for each method (Table 6). While the number of distinct clusters of actives found in the top 10 molecules is somewhat comparable for ACPC, Pharao, MACCS and Shape-it; the rate at which new clusters of actives are discovered by ACPC doesn't increase as fast as other methods. But this is to be expected; pharmacophores (Pharao) are a powerful way of generalizing atom types while shape (Shape-it) is an even more permissive representation of molecules.

Early during the development of the method, scoring the similarity of descriptors with Pearson's r and Spearman's rank order correlation coefficient [52] was tried. Later on, distance metrics for continuous variables were tried (some equations can be found in [26,44]): the Tanimoto coefficient, Tversky_ref, Tversky_db and $\frac{1}{1+d}$ where d is the Manhattan distance or the Euclidean distance. None of these performed better than cross-correlation in tests on small random partitions of the DUD (20 queries chosen randomly across all ligands and targets).

ACPC's good performance might be explained by the fact that all atoms of a molecule are considered at the same time and all intra molecular distances are handled in the same way. While atom centers partially encode the shape, partial charges encode some of the recognition features (eg. hydrogen bond acceptors and donors). ACPC measures the global similarity of two molecules in terms of intra molecular vectors and partial charges. The high number of molecules per second the method can process

Table 4 Median AUCs on the "1conf" dataset

Target (L/D)	ACPC	Pharao	MACCS	Shape-it
ace (49/1753)	0.47	0.55	***0.75***	0.48
ache (106/3711)	***0.80***	0.56	0.67	0.63
ada (37/844)	***0.81***	0.65	***0.81***	0.62
alr2 (23/939)	0.61	0.61	***0.70***	0.63
ampc (21/767)	0.88	0.87	***0.90***	0.81
ar (74/2709)	0.73	0.66	0.72	***0.79***
cdk2 (58/1866)	***0.90***	0.62	0.55	0.60
comt (10/425)	0.82	0.85	***0.90***	0.40
cox1 (25/885)	0.29	0.42	***0.55***	0.46
cox2 (411/12281)	***0.94***	0.92	0.85	0.90
dhfr (405/7418)	***1.00***	0.87	0.95	0.62
egfr (458/14449)	0.92	***0.94***	0.83	0.54
er+ (67/2387)	***0.95***	0.76	***0.95***	0.77
er- (39/1330)	0.59	0.70	***0.91***	0.67
fgfr1 (120/4305)	***0.93***	0.84	0.81	0.53
fxa (146/4969)	***0.94***	0.79	0.85	0.72
gart (31/845)	***1.00***	0.86	0.98	0.68
gpb (50/2072)	***0.99***	0.95	0.89	0.62
gr (78/2803)	0.75	0.77	***0.83***	0.69
hivpr (57/1797)	0.70	0.72	0.60	***0.80***
hivrt (41/1433)	***0.86***	0.51	0.47	0.65
hmga (35/1362)	0.90	0.74	***0.95***	0.81
hsp90 (25/918)	0.62	***0.85***	0.77	0.76
inha (79/3131)	0.60	0.59	***0.70***	0.51
mr (14/561)	0.77	0.71	***0.87***	***0.87***
na (49/1826)	***0.94***	0.68	0.91	0.59
p38 (366/8722)	0.55	***0.90***	0.72	0.66
parp (35/1296)	***1.00***	0.92	0.90	0.60
pde5 (76/1955)	0.66	***0.71***	0.64	0.60
pdgfrb (169/5560)	0.50	***0.72***	0.59	0.57
pnp (30/962)	***0.99***	0.93	0.87	0.59
pparγ (82/2635)	0.91	0.88	***0.92***	0.83
pr (27/989)	***0.81***	0.76	0.73	0.57
rxrα (20/724)	0.85	***1.00***	0.99	0.62
sahh (32/1250)	***0.99***	0.74	0.96	0.78
src (159/5904)	***0.91***	0.80	0.74	0.53
thrombin (67/2308)	***0.86***	0.79	0.77	0.69
tk (22/860)	***0.99***	0.90	0.91	0.70
trypsin (46/1565)	0.88	***0.90***	0.78	0.73
vegfr2 (77/2701)	***0.81***	0.66	0.53	0.50
Average	0.81	0.77	0.79	0.65
Median	0.86	0.77	0.82	0.63
|Best method|	20	7	13	3

For each of the 40 targets, each ligand in the ligands list was used in turn as the query. On each line, the maximum value is underlined and in bold font.
L = number of ligands; D = number of decoys.

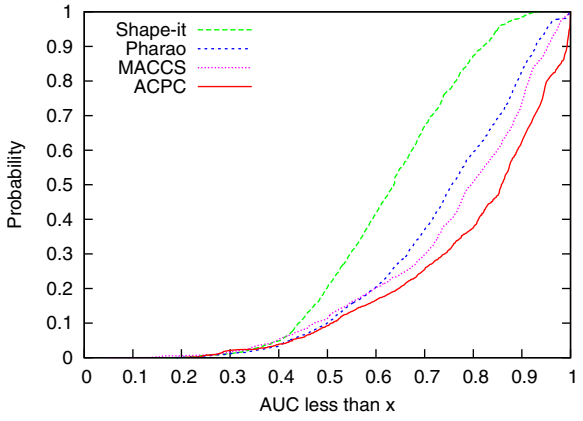


Figure 4 Cumulative distribution functions for Shape-it, Pharao, MACCS and ACPC on all queries of the "1conf" dataset.

is a direct benefit from its simplicity and reliance on a strong mathematical property: being rotation-translation invariant. In ACPC, molecules do not need to be optimally superposed before scoring and the electrostatic potential field is not computed.

Recommended usage

ACPC was designed to be rotation and translation invariant. However, ACPC is not invariant to the conformer of a molecule, neither to the charge model that was used to assign partial charges (Table 2). ACPC is also sensitive to the choice of the *dx* parameter (Table 1). Hence, the following protocol has been validated: the query molecule(s) *and* the database to screen must be prepared in the same way. The same software with same parameters must be used to assign partial charges and generate conformers for *all* molecules.

Limitations

Our method, by construction, cannot distinguish a molecule m from its enantiomer. If n is a perfect mirror image of m and m' is a copy of m with all partial charges reversed (sign flipped) and n' is a copy of n with all partial charges reversed, then they cannot be distinguished.

$$CC(m,m,dx) = CC(m,m',dx) = CC(m,n,dx) = CC(m,n',dx)$$

Also, since the method is purely based on partial charges, it should perform poorly if the recognition of a binding site by a query ligand is not mostly driven by electrostatics. This should be the case for binding sites that are predominantly non-polar.

While release two of the DUD [35] was used to prepare datasets, it has known shortcomings. DUD was initially created as a benchmark set for molecular docking but, like

Table 5 Average AUCs on the "25conf" dataset

target (L/D)	$ACPC_{1c}$	$ACPC_{25c}$	ACPC	Pharao	MACCS	Shape-it	ES3DH	ES3D	Gpi3	pi3	pi4	TAD	TAT	TGD	TGT
ace (1140/43653)	0.49	0.48	0.45	0.58	0.68	0.57	0.34	0.44	***0.72***	0.60	0.62	0.61	0.57	0.49	0.60
ache (2450/89138)	0.80	0.80	0.80	0.69	0.72	0.69	0.48	0.28	0.81	***0.85***	***0.85***	0.66	0.75	0.60	0.74
ada (876/18697)	0.82	0.81	0.78	0.66	0.80	0.64	0.71	0.42	0.81	0.75	0.74	0.84	0.82	0.85	***0.87***
alr2 (348/16063)	0.60	0.59	0.47	***0.74***	0.42	0.52	0.35	0.33	0.54	0.66	0.64	0.32	0.24	0.29	0.44
ampc (507/15069)	0.84	0.83	0.89	0.86	0.88	0.71	0.82	0.59	0.87	0.87	***0.90***	0.75	0.83	0.84	0.75
ar (363/42557)	0.75	0.73	0.72	0.67	0.80	0.75	0.68	0.44	0.71	0.76	0.71	***0.81***	0.66	0.80	0.67
cdk2 (1138/43682)	0.88	0.88	***0.93***	0.62	0.49	0.54	0.42	0.50	0.66	0.68	0.67	0.34	0.41	0.34	0.46
comt (124/7148)	0.83	0.81	0.80	***0.81***	0.77	0.65	0.38	0.34	0.76	0.62	0.75	0.43	0.46	0.46	0.53
cox1 (422/15672)	0.30	0.29	0.27	0.55	0.51	0.46	0.47	0.38	0.50	0.40	0.48	0.49	0.61	0.50	***0.64***
cox2 (6483/270311)	0.94	0.94	***0.87***	0.46	0.15	0.16	0.62	0.52	0.22	0.26	0.33	0.49	0.50	0.65	0.74
dhfr (9550/166944)	1.00	1.00	***1.00***	0.94	0.96	0.56	0.63	0.46	0.98	0.93	0.98	0.99	***1.00***	0.98	0.91
egfr (9964/337283)	0.91	0.89	0.90	***0.96***	0.78	0.57	0.49	0.25	0.89	0.69	0.89	0.50	0.66	0.46	0.53
er+ (289/39507)	0.92	0.91	0.94	0.81	0.95	0.63	0.26	0.38	0.76	0.89	0.68	***0.96***	***0.96***	***0.96***	***0.96***
er- (975/32961)	0.62	0.60	0.71	0.75	0.89	0.66	0.29	0.40	0.83	0.84	0.69	***0.95***	0.92	***0.95***	0.92
fgfr1 (2360/106612)	0.93	0.94	***0.87***	0.54	0.55	0.63	0.61	0.41	0.79	0.59	0.49	0.37	0.49	0.36	0.50
fxa (3647/123379)	0.89	0.89	***0.64***	0.61	0.48	0.40	0.39	0.42	0.42	0.40	0.44	0.24	0.42	0.29	0.30
gart (775/20938)	1.00	1.00	***1.00***	0.94	0.98	0.69	0.89	0.89	0.97	0.97	0.98	0.98	0.98	0.96	0.98
gpb (845/44604)	0.97	0.96	***0.87***	0.23	0.68	0.42	0.24	0.29	0.34	0.16	0.32	0.18	0.20	0.17	0.16
gr (553/56086)	0.73	0.72	0.75	0.75	***0.86***	0.55	0.50	0.48	0.79	0.82	0.85	0.64	0.67	0.62	0.60
hivpr (1404/44909)	0.72	0.72	0.79	0.73	0.47	0.74	0.63	0.56	0.44	0.42	0.53	0.92	***0.93***	0.90	0.92
hivrt (822/32688)	0.86	0.86	***0.82***	0.47	0.43	0.28	0.55	0.47	0.47	0.57	0.51	0.31	0.39	0.39	0.34
hmga (814/33684)	0.89	0.89	0.87	0.83	0.94	0.85	0.57	0.53	0.94	0.93	0.94	***0.99***	0.98	0.97	0.98
hsp90 (572/20983)	0.58	0.56	0.67	0.81	0.70	0.58	0.84	0.59	***0.93***	0.89	0.79	0.52	0.66	0.58	0.61
inha (1782/71064)	0.58	0.58	0.59	0.48	***0.78***	0.48	0.24	0.30	0.33	0.35	0.36	0.41	0.66	0.49	0.71
mr (79/10177)	0.73	0.70	0.54	0.18	0.69	0.72	***0.82***	0.60	0.75	0.65	0.48	0.49	0.43	0.46	0.58
na (987/44278)	0.92	0.92	***0.93***	0.79	0.91	0.39	0.37	0.33	0.87	0.79	0.84	0.72	0.88	0.74	0.72
p38 (7014/186992)	0.54	0.51	0.57	***0.89***	0.76	0.69	0.62	0.38	0.87	0.82	0.88	0.48	0.60	0.38	0.59
parp (245/12871)	0.97	0.97	***0.99***	0.93	0.93	0.73	0.55	0.53	0.97	0.93	0.96	0.83	0.89	0.74	0.89
pde5 (1751/48431)	0.66	0.65	***0.75***	0.67	0.68	0.69	0.39	0.44	0.67	0.62	0.67	0.31	0.35	0.36	0.34
pdgfrb (3206/134591)	0.51	0.52	0.49	0.46	0.49	0.61	0.57	0.49	***0.62***	0.44	0.42	0.34	0.46	0.36	0.50
pnp (560/16694)	0.99	0.98	***0.99***	0.95	0.90	0.63	0.36	0.52	0.90	0.76	0.91	0.82	0.86	0.87	0.91
pparγ (2010/65783)	0.91	0.90	0.88	0.89	0.89	0.70	0.21	0.25	0.92	**0.94**	0.92	0.86	***0.94***	0.76	0.86

Table 5 Average AUCs on the "25conf" dataset *Continued*

pr (178/15966)	0.70	0.70	0.37	0.37	0.48	0.67	0.30	0.19	0.73	***0.76***	0.56	0.61	0.66	0.64	0.63
rxrα (392/17243)	0.77	0.77	0.86	***1.00***	0.99	0.70	0.53	0.49	0.95	0.97	0.99	0.94	0.95	0.99	0.90
sahh (586/25622)	0.98	0.98	0.96	0.57	***0.97***	0.71	0.78	0.65	0.96	0.93	0.95	0.89	0.95	0.90	0.96
src (2945/145751)	0.90	0.90	***0.81***	0.51	0.47	0.65	0.69	0.53	0.60	0.43	0.36	0.30	0.41	0.26	0.40
thrombin (1576/57564)	0.87	0.88	***0.95***	0.74	0.51	0.65	0.54	0.58	0.65	0.48	0.36	0.71	0.68	0.62	0.66
tk (379/15017)	0.98	0.98	***0.98***	0.84	0.92	0.54	0.47	0.37	0.89	0.86	0.87	0.86	0.86	0.85	0.89
trypsin (1128/39065)	0.90	0.90	***0.98***	0.64	0.29	0.45	0.46	0.28	0.30	0.29	0.41	0.72	0.80	0.69	0.81
vegfr2 (1604/66098)	0.79	0.78	***0.67***	0.47	0.42	0.44	0.32	0.24	0.57	0.54	0.48	0.37	0.41	0.33	0.38
Average	0.80	0.79	0.78	0.68	0.70	0.59	0.51	0.44	0.72	0.68	0.68	0.62	0.67	0.62	0.67
Median	0.85	0.84	0.81	0.71	0.74	0.63	0.49	0.44	0.76	0.72	0.69	0.62	0.66	0.62	0.67
|Best method|	N/A	N/A	17	5	3	0	1	0	3	3	2	4	4	2	3

For each of the 40 targets, the query was the last ligand in the ligands list of each target. For each target, the maximum AUC reached is underlined and in bold font. L = number of ligands; D = number of decoys. The $ACPC_{1c}$ and $ACPC_{25c}$ columns were computed and added for comparison. $ACPC_{1c}$ (resp. $ACPC_{25c}$) shows the average AUC reached when using all active ligands as queries for ACPC on "1conf" (resp. "25conf"). Their |best method| cells were not filled in since they concern different experiments than other columns.

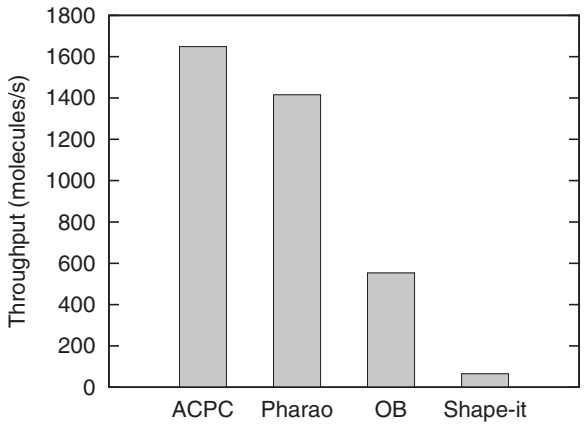


Figure 5 Average processing speed of ACPC, Pharao, Open Babel (OB) and Shape-it on the target with the most ligands and decoys (egfr).

some previous authors [19,53,54], we use it to test ligand-based virtual screening methods. If some ligands L1 and L2 of the same target are targeting different sub-pockets of the binding site, it is incorrect to use one of them (in a ligand-based approach) to find the other. Another problem previously noticed by other authors [55] is that DUD decoys are only supposed to be inactive. If tested experimentally, some decoys may be found to be active. In future studies, DUD Enhanced, a more recent version of DUD [56] which overcomes some of its previous drawbacks and includes more targets might be used.

Potential users should keep in mind that there is no silver bullet for *in-silico* drug discovery. ACPC is no exception. From previously shown results, MACCS or Pharao or some of the MOE fingerprints are seen to perform better than ACPC on several targets. Also, Pharao, MACCS and Shape-it seem to promote scaffold diversity of actives earlier in the ranked list of compounds (Table 6).

Upcoming features

The following features are under consideration for future releases of ACPC. On the purely technical side, a GPU-based version of the tool is doable [57] to reach higher throughput. Combined with a metric distance, clustering compounds databases would then become computationally tractable. Other interesting topics would require more research and experiments, such as the automatic creation of a consensus query from a set of known actives, investigating other orthogonal feature spaces such as atomic radii, solvent accessible areas and per atom hydrophobic contribution, to increase the discriminative power of the method. The choice and parametrization of suitable kernel functions for use in Kernel Density Estimates (KDE) is also an interesting direction that could result in reduced sensitivity to the discretization parameter and probably better AUCs (at the cost of heavier computation). With KDE, the method could probably be extended to cluster binding sites based on the partial charges of their surface atoms.

Conclusions

We revisited the family of rotation-translation invariant molecular descriptors and proposed a new, simple but powerful encoding of the autocorrelation of partial charges of a 3D molecule. Our implementation is fast and displays good performance in retrospective ligand-based virtual screening experiments. ACPC should be a useful tool for ligand-based virtual screening. ACPC is open source, freely available and automatically installable as an OPAM [58] package. Requests and contributions from users are welcome.

Availability and requirements

Project Name: ACPC
Project home page: http://www.riken.jp/zhangiru/software.html
Operating system: Linux
Programming language: OCaml http://ocaml.org
Other requirements: the OCaml Package Manager (OPAM) http://opam.ocaml.org/
Dataset (352 MB): http://www.riken.jp/zhangiru/software/DUD_ACPC_1.0_validation.tar.xz The CROC Python package (optional) http://pypi.python.org/pypi/CROC/ Gnuplot [34] (optional) http://www.gnuplot.info/
License: BSD
Any restrictions to use by non-academics: None

Table 6 Average number of distinct clusters found for active molecules among the top N ranked molecules for cox2 and egfr in "1conf" using 10 random queries

Target	cox2				egfr			
N	ACPC	Pharao	MACCS	Shape-it	ACPC	Pharao	MACCS	Shapeit
10	2.3	2.2	2.8	2.3	0.9	2.4	1.4	2.2
20	2.7	3.3	3.9	3.5	1.1	3.0	2.4	3.1
30	3.1	4.4	4.5	4.6	1.3	4.4	2.9	3.6
40	3.4	4.9	5.1	5.4	1.5	5.2	3.5	3.9
50	3.5	5.8	5.4	6.3	2.0	5.8	4.7	4.1

Endnotes

[a]Receiver Operating Characteristic.
[b]They should have the same molecule name.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

FB designed the method, wrote the software, ran experiments and prepared figures and tables. XYL, FB and AV prepared the datasets. FB, AV, XYL and KYJZ analyzed the results, wrote, read and approved the final manuscript.

Acknowledgments

We thank people who release open source software and open datasets: they enable and accelerate our research significantly. We thank Prof. Jeremy Tame for reading the manuscript. We thank the MOE support team in Japan for MOE batch scripts. We acknowledge the use of MarvinView 6.1.2 (2013) from ChemAxon (http://www.chemaxon.com) to view chemical structures. This work was supported by the "Initiative Research Unit" program from RIKEN (Japan) and the Japanese Society for the Promotion of Science (JSPS). We thank RIKEN for an allocation of computing resources on the RIKEN Integrated Cluster of Clusters (RICC).

Received: 8 January 2014 Accepted: 22 April 2014
Published: 10 May 2014

References

1. Bender A, Glen RC: **Molecular similarity: a key technique in molecular informatics.** *Org Biomol Chem* 2004, **2:**3204–3218.
2. Bender A, Jenkins JL, Scheiber J, Sukuru SCK, Glick M, Davies JW: **How similar are similarity searching methods? A principal component analysis of molecular descriptor space.** *J Chem Inf Model* 2009, **49**(1):108–119.
3. Wegner JK, Sterling A, Guha R, Bender A, Faulon J-L, Hastings J, O'Boyle N, Overington J, Van Vlijmen H, Willighagen E: **Cheminformatics.** *Commun ACM* 2012, **55**(11):65–75.
4. Kubinyi H: **Similarity and dissimilarity: a medicinal chemist's view.** *Perspect Drug Discov Des* 1998, **9–11**(0):225–252.
5. Willett P: **Similarity-based virtual screening using 2D fingerprints.** *Drug Discov Today* 2006, **11**(23):1046–1053.
6. Eckert H, Bajorath J: **Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches.** *Drug Discov Today* 2007, **12**(5):225–233.
7. Hu G, Kuang G, Xiao W, Li W, Liu G, Tang Y: **Performance evaluation of 2D fingerprint and 3D shape similarity methods in virtual screening.** *J Chem Inf Model* 2012, **52**(5):1103–1113.
8. Riniker S, Landrum G: **Open-source platform to benchmark fingerprints for ligand-based virtual screening.** *J Cheminformatics* 2013, **5**(1):26.
9. Teixeira AL, Falcao AO: **Noncontiguous Atom Matching Structural Similarity Function.** *J Chem Inf Model* 2013, **53**(10):2511–2524.
10. Todeschini R, Consonni V: *Molecular Descriptors for Chemoinformatics (2 Volumes), vol. 41*. Weinheim: Wiley-VCH; 2009.
11. Raymond J, Willett P: **Maximum common subgraph isomorphism algorithms for the matching of chemical structures.** *J Comput Aided Mol Des* 2002, **16**(7):521–533.
12. Grosso A, Locatelli M, Pullan W: **Simple ingredients leading to very efficient heuristics for the maximum clique problem.** *J Heuristics* 2008, **14**(6):587–612.
13. Kawabata T: **Build-up algorithm for atomic correspondence between chemical structures.** *J Chem Inf Model* 2011, **51**(8):1775–1787.
14. Mauri A, Consonni V, Pavan M, Todeschini R: **Dragon software: An easy approach to molecular descriptor calculations.** *MATCH Commun Math Comput Chem* 2006, **56:**237–248.
15. Naray-Szabo G: **Analysis of molecular recognition: steric electrostatic and hydrophobic complementarity.** *J Mol Recognit* 1993, **6**(4):205–210.
16. Muchmore SW, Souers AJ, Akritopoulou-Zanze I: **The use of three-dimensional shape and electrostatic similarity searching in the identification of a melanin-concentrating hormone receptor 1 antagonist.** *Chem Biol Drug Des* 2006, **67**(2):174–176.
17. Naylor E, Arredouani A, Vasudevan SR, Lewis AM, Parkesh R, Mizote A, Rosen D, Thomas JM, Izumi M, Ganesan A, Galione A, Churchill GC: **Identification of a chemical probe for NAADP by virtual screening.** *Nat Chem Biol* 2009, **5**(5):220–226.
18. OpenEye Scientific Software I: *OEChem 1.7.4*. Santa Fe: OpenEye Scientific Software, Inc; 2010.
19. Armstrong MS, Morris GM, Finn PW, Sharma R, Moretti L, Cooper RI, Richards WG: **ElectroShape: fast molecular similarity calculations incorporating shape, chirality and electrostatics.** *J Comput Aided Mol Des* 2010, **24**(9):789–801.
20. Voet A, Berenger F, Zhang KYJ: **Electrostatic similarities between protein and small molecule ligands facilitate the design of protein-protein interaction inhibitors.** *PLoS ONE* 2013, **8**(10):75762.
21. Moreau G, Broto P: **The autocorrelation of a topological structure: a new molecular descriptor.** *New J Chem* 1980, **4**(6):359–360.
22. Carhart RE, Smith DH, Venkataraghavan R: **Atom pairs as molecular features in structure-activity studies: definition and applications.** *J Chem Inf Comput Sci* 1985, **25**(2):64–73.
23. Wagener M, Sadowski J, Gasteiger J: **Autocorrelation of molecular surface properties for modeling corticosteroid binding globulin and cytosolic ah receptor activity by neural networks.** *J Am Chem Soc* 1995, **117**(29):7769–7775.
24. Anzali S, Barnickel G, Krug M, Sadowski J, Wagener M, Gasteiger J, Polanski J: **The comparison of geometric and electronic properties of molecular surfaces by neural networks: Application to the analysis of corticosteroid-binding globulin activity of steroids.** *J Comput Aided Mol Des* 1996, **10**(6):521–534.
25. Bauknecht H, Zell A, Bayer H, Levi P, Wagener M, Sadowski J, Gasteiger J: **Locating biologically active compounds in medium-sized heterogeneous datasets by topological autocorrelation vectors: Dopamine and benzodiazepine agonists.** *J Chem Inf Comput Sci* 1996, **36**(6):1205–1213.
26. Fechner U, Franke L, Renner S, Schneider P, Schneider G: **Comparison of correlation vector methods for ligand-based similarity searching.** *J Comput Aided Mol Des* 2003, **17**(10):687–698.
27. Moro S, Bacilieri M, Cacciari B, Spalluto G: **Autocorrelation of molecular electrostatic potential surface properties combined with partial least squares analysis as new strategy for the prediction of the activity of human A3 adenosine receptor antagonists.** *J Med Chem* 2005, **48**(18):5698–5704. PMID: 16134938.
28. Broto P, Moreau G, Vandycke C: **Molecular structures: perception, autocorrelation descriptor and SAR studies. Autocorrelation descriptor.** *Eur J Med Chem* 1984, **19**(1):66–70.
29. Broto P, Moreau G, Vandycke C: **Molecular structures: perception, autocorrelation descriptor and SAR studies. Use of the autocorrelation descriptors in the QSAR study of two non-narcotic analgesic series.** *Eur J Med Chem* 1984, **19**(1):79–84.
30. Schneider G, Neidhart W, Giller T, Schmid G: **"scaffold-hopping" by topological pharmacophore search: a contribution to virtual screening.** *Angew Chem Int Ed* 1999, **38**(19):2894–2896.
31. Wand MP: **Fast Computation of multivariate kernel estimators.** *J Comput Graph Stat* 1994, **3**(4):433–445.
32. Swamidass SJ, Azencott C-A, Daily K, Baldi P: **A CROC stronger than ROC: measuring, visualizing and optimizing early retrieval.** *Bioinformatics* 2010, **26**(10):1348–1356.
33. Danelutto M, Cosmo RD: **A minimal disruption skeleton experiment: seamless map and reduce embedding in OCaml.** *Procedia Comput Sci* 2012, **9**(0):1837–1846. Proceedings of the International Conference on Computational Science, ICCS 2012.
34. Janert PK: *Gnuplot in Action: Understanding Data with Graphs*. Greenwich: Manning Publications Co.; 2009.
35. Huang N, Shoichet BK, Irwin JJ: **Benchmarking sets for molecular docking.** *J Med Chem* 2006, **49**(23):6789–6801.
36. Hawkins PCD, Nicholls A: **Conformer generation with OMEGA: learning from the data set and the analysis of failures.** *J Chem Inf Model* 2012, **52**(11):2919–2936.
37. O'Boyle N, Banck M, James C, Morley C, Vandermeersch T, Hutchison G: **Open Babel: an open chemical toolbox.** *J Cheminformatics* 2011, **3**(1):33.
38. Inc. CCG: *Molecular Operating Environment (MOE), 2011.10 edn*. Montreal: Chemical Computing Group Inc.; 2011.

39. Gasteiger J, Marsili M: **Iterative partial equalization of orbital electronegativity a rapid access to atomic charges.** *Tetrahedron* 1980, **36**(22):3219–3228.
40. Chen J, Martinez TJ: **QTPIE: Charge transfer with polarization current equalization. a fluctuating charge model with correct asymptotics.** *Chem Phys Lett* 2007, **438**(4):315–320.
41. Rappe AK, Goddard WA: **Charge equilibration for molecular dynamics simulations.** *J Phys Chem* 1991, **95**(8):3358–3363.
42. Halgren TA: **Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94.** *J Comput Chem* 1996, **17**(5):490–519.
43. Halgren TA, Nachbar RB: **Merck molecular force field. IV. conformational energies and geometries for MMFF94.** *J Comput Chem* 1996, **17**(5):587–615.
44. Taminau J, Thijs G, Winter HD: **Pharao: Pharmacophore alignment and optimization.** *J Mol Graph Model* 2008, **27**(2):161–169.
45. **Silicos-it company: chemoinformatics services and software.** [http://silicos-it.com/index.html]
46. **The Shape-it software from Silicos-it.** [http://silicos-it.com/software/shape-it/1.0.1/shape-it.html]
47. Grant JA, Gallardo MA, Pickup BT: **A fast method of molecular shape comparison: a simple application of a Gaussian description of molecular shape.** *J Comput Chem* 1996, **17**(14):1653–1666.
48. Brown RD, Martin YC: **Use of structure-activity data to compare structure-based clustering methods and descriptors for use in compound selection.** *J Chem Inf Comput Sci* 1996, **36**(3):572–584.
49. Kearsley SK, Sallamack S, Fluder EM, Andose JD, Mosley RT, Sheridan RP: **Chemical similarity using physiochemical property descriptors†.** *J Chem Inf Comput Sci* 1996, **36**(1):118–127.
50. Berenger F, Coti C, Zhang KYJ: **PAR: a PARallel and distributed job crusher.** *Bioinformatics* 2010, **26**(22):2918–2919.
51. Barker EJ, Gardiner EJ, Gillet VJ, Kitts P, Morris J: **Further development of reduced graphs for identifying bioactive compounds.** *J Chem Inf Comput Sci* 2003, **43**(2):346–356.
52. Press WH, Teukolsky SA, Vetterling WT, Flannery BP: *Numerical Recipes 3rd Edition: The Art of Scientific Computing, 3rd edn.* New York: Cambridge University Press; 2007.
53. von Korff M, Freyss J, Sander T: **Comparison of Ligand- and Structure-Based Virtual Screening on the DUD Data Set.** *J Chem Inf Model* 2009, **49**(2):209–231.
54. Venkatraman V, Perez-Nueno VI, Mavridis L, Ritchie DW: **Comprehensive Comparison of Ligand-Based Virtual Screening Tools Against the DUD Data set Reveals Limitations of Current 3D Methods.** *J Chem Inf Model* 2010, **50**(12):2079–2093.
55. Vogel SM, Bauer MR, Boeckler FM: **DEKOIS: Demanding Evaluation Kits for Objective in Silico screening a versatile tool for benchmarking docking programs and scoring functions.** *J Chem Inf Model* 2011, **51**(10):2650–2665.
56. Mysinger MM, Carchia M, Irwin JJ, Shoichet BK: **Directory of Useful Decoys, Enhanced (DUD-E): better ligands and decoys for better benchmarking.** *J Med Chem* 2012, **55**(14):6582–6594.
57. Bourgoin M, Chailloux E, Lamotte J-L: **SPOC: GPGPU programming through stream processing with OCaml.** *Parallel Process Lett* 2012, **22**(2):1240007.
58. **The OCaml Package Manager (OPAM).** [http://opam.ocaml.org/]

doi:10.1186/1758-2946-6-23
Cite this article as: Berenger *et al.*: A rotation-translation invariant molecular descriptor of partial charges and its use in ligand-based virtual screening. *Journal of Cheminformatics* 2014 **6**:23.

Publish with ChemistryCentral and every scientist can read your work free of charge

"Open access provides opportunities to our colleagues in other parts of the globe, by allowing anyone to view the content free of charge."

W. Jeffery Hurst, The Hershey Company.

- available free of charge to the entire scientific community
- peer reviewed and published immediately upon acceptance
- cited in PubMed and archived on PubMed Central
- yours — you keep the copyright

Submit your manuscript here:
http://www.chemistrycentral.com/manuscript/

ChemistryCentral

Chapitre 4

Résultats complémentaires

Dans ce chapitre, nous présentons des résultats qui n'ont pas fait l'objet d'une publication dans un journal revu par des pairs. Il peut s'agir de résultats présentés sur un poster lors d'une conférence ou d'outils non encore publiés mais qui sont déjà utilisés par des collègues.

Dans la section qui suit, nous montrons un autre emploi de l'inégalité triangulaire du RMSD. Cette fois il s'agit non pas de faire du clustering mais de cribler rapidement un ensemble de fragments de protéines.

4.1 Sélection rapide de fragments de protéines

De nos jours, un grand nombre de structures de protéines sont disponibles (109479 en Avril 2016 sur `www.pdb.org`[16]). Certaines activités de conception ou de modélisation de protéines requièrent d'interroger l'intégralité de la PDB avec un fragment de protéine. La longueur de ce fragment requête peut varier et même contenir des trous. Pour certaines applications, il peut être préférable de travailler sur un sous-ensemble spécifique de la PDB mais aussi avec des structures non publiées. Des logiciels de prédiction de structure tels que Rosetta[91], QUARK[180] et EdaFold[141] utilisent des fragments comme briques de construction des modèles de protéines. Les fragments sont aussi utilisés pour résoudre le problème de la phase en cristallographie[125,140] et afin de reconstruire des modèles[1]. La qualité des modèles de protéines peut être améliorée en combinant des fragments avec de la dynamique moléculaire[182]. D'autres applications incluent la reconstruction des boucles

manquantes en cristallographie[92,136], la greffe de boucle sur une protéine existante ainsi que d'autres méthodes d'ingénierie de protéines[24,161].

Plusieurs cueilleurs de fragments[25,57,60] existent déjà, ainsi que des bases de données de fragments[21,165]. Le logiciel Super[25] est particulièrement intéressant. Super utilise la borne basse du RMSD[158] afin de filtrer rapidement la totalité de l'espace des fragments. Le logiciel BCSearch[60] est aussi intéressant. Il utilise une méthode cinquante fois plus rapide a calculer que le RMSD pour mesurer la similarité entre fragments[59]. Cependant, comme nos collègues faisaient à la fois de la conception de protéines et du raffinement de modèles pour la cristallographie, nous avons dû créer notre propre cueilleur de fragments avec des options spécifiques.

Au contraire des logiciels existants qui utilisent le C_αRMSD, Fragger utilise le RMSD tous atomes du squelette peptidique (N,C_α, C, O) afin de préserver le détail des structures secondaires. Fragger supporte n'importe quelle taille de fragment et peut utiliser tout ensemble de fichiers PDB pour créer un ensemble de fragments (que l'on peut considérer comme une base de données). Fragger peut chercher de manière efficace cet ensemble, étant donné un fragment requête et une tolérance maximale (un RMSD en Å) par rapport à ce fragment. Les fragments qui répondent à la requête sont triés par ordre croissant en fonction de leur distance au fragment requête. Le fragment requête peut contenir des trous et la séquence d'acides aminés autorisés dans les réponses peut être contrainte via une expression formelle composée de codes d'acides aminés à une lettre. On peut aussi ne considérer que certains acides aminés du fragment requête, par exemple les trois premiers et les trois derniers AA d'une boucle. Fragger incorpore aussi un outil pour calculer le RMSD d'un fragment contre beaucoup.

Fragger exploite l'inégalité triangulaire du RMSD[146] afin d'élaguer l'espace des fragments (figure 4.1 et algorithme 4). Fragger calcule le RMSD en utilisant la méthode QCP[154]. Fragger est écrit en OCaml, sauf le calcul du RMSD qui est écrit en C++ dans une version améliorée de l'outil ranker de Durandal[14]. Les calculs sont parallélisés avec la librairie Parmap[29] sur les processeurs multi-cœurs.

Fragger propose des fonctionnalités très spécifiques : écrire en sortie seulement les N fragments les plus proches de la requête, voire même les N premiers fragments trouvés

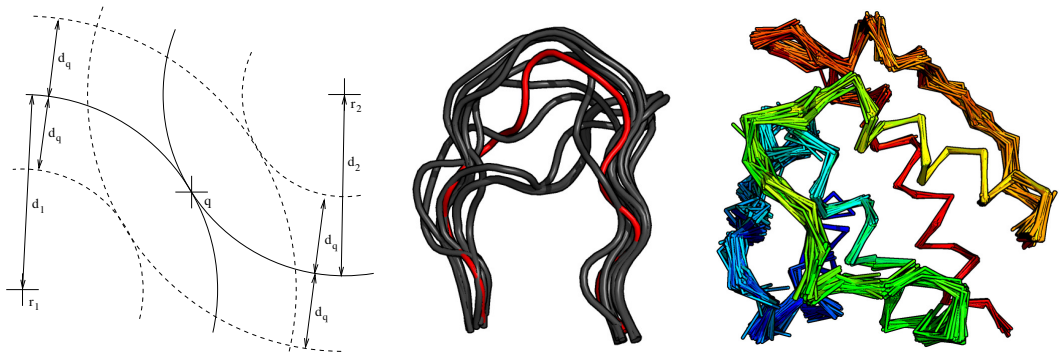


FIGURE 4.1 – **À gauche :** élagage de l'espace des fragments pour la tolérance d_q et le fragment requête q. q est à la distance d_1 (resp. d_2) du fragment de référence r_1 (resp. r_2). Seuls les fragments qui sont à la fois à la distance $d_1 \pm d_q$ de r_1 et $d_2 \pm d_q$ de r_2 donneront lieu à un calcul de RMSD. **Au milieu :** boucles de 13 résidus qui peuvent connecter les résidus ALA 98 à GLY 110 de la chaîne A de la protéine PDB:1MEL. Seuls les trois premiers et derniers acides aminés de la boucle requête (colorée en rouge) ont été utilisés afin d'ordonner les fragments qui répondent à la requête. **À droite :** squelette de la protéine PDB:1BKR, recouvert par des fragments longs de 10 amino acides, provenant tous de protéines non homologues.

Algorithm 4 Requête utilisant un fragment de protéine q et une distance d_q

Input: db : ensemble de fragments à interroger
Input: r : ensemble des fragments références (des index)
Input: q : fragment requête
Input: d_q : distance maximale au fragment requête (tolérance)
Output: mf : ensemble des fragments qui répondent à la requête

```
mf ← db
{Requête approximative : élagage de l'espace de recherche}
for r_j in r do
    d ← distance(q, r_j)
    d_inf ← d − d_q
    d_sup ← d + d_q
    mf ← {∀f_i ∈ mf | distance(f_i, r_j) ∈ [d_inf, d_sup]}
    {distance(f_i, r_j) provient de l'index}
end for
{Requête exacte : raffinement de la requête approximative}
mf ← {∀f_i ∈ mf | distance(f_i, q) ≤ d_q}
return mf
```

(beaucoup plus rapide dans certains cas). Fragger peut lire et écrire les fragments au format PDB, mais aussi dans un format binaire qui est plus compact, rapide à lire et à écrire. En plus de contraindre les séquences autorisées à répondre à une requête, Fragger peut exclure une liste de codes PDB dans les réponses et varier automatiquement la tolérance RMSD jusqu'à ce qu'un nombre suffisant de fragments ait été trouvé.

Des tests de performance ont été menés en utilisant un seul cœur d'une machine avec un processeur Intel Xeon à 2.4GHz. La machine possède 12GB de RAM et utilise Ubuntu Linux 12.04. L'ensemble des PDB utilisés se compose de toutes les protéines déterminées par cristallographie à rayons X et sans homologues (30% de similarité de séquence au maximum). Cet ensemble contient 13554 structures. Les expériences montrent que Fragger est rapide même quand on travaille à l'échelle de la PDB toute entière. Une requête prend entre cinq et sept secondes. Ce temps varie en fonction du fragment requête, des fragments de référence ainsi que de la tolérance RMSD à la requête. Pour certaines tâches, il n'est pas nécessaire de créer des index RMSD avec des fragments de référence car la génération de fragments à la volée ainsi que les calculs de RMSD sont suffisamment rapides. Par exemple, il faut seulement 15 secondes pour générer tous les fragments de 13 AA qui commencent par ALA et se terminent par GLY (il y en a 41200). Les ordonner par rapport à leur distance au fragment requête prend seulement 1.5s. Quand il travaille sur des fichiers PDB, l'outil ranker inclus dans Fragger peut calculer 66580 (resp. 23784) RMSD/s pour des fragments de longueur trois (resp. neuf) AA. Ces chiffres montent à 304149 (resp. 138744) RMSD/s quand Fragger utilise son format interne binaire d'encodage des PDB.

Dans la section qui suit, nous montrons plusieurs améliorations et fonctionnalités qui ont été ajoutées au logiciel ACPC après sa publication.

4.2 ACPC : fonctionnalités avancées pour le LBVS

Après la publication de l'article sur ACPC, nous avons continué à travailler dessus afin de présenter un poster lors de l'école d'été de chémoinformatique à Strasbourg en 2014. ACPC-1.2 permet de travailler dans d'autres espaces que celui électrostatique, permet les requêtes consensus, peut annoter la molécule requête afin de donner un retour visuel aux

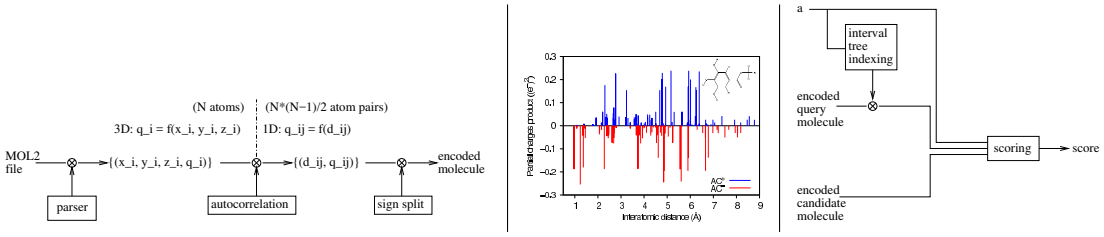


FIGURE 4.2 – **À gauche :** encodage de la molécule. **Au milieu :** valeurs positives (en bleu) et négatives (en rouge) de l'autocorrélation des charges partielles de la molécule montrée en 2D. **À droite :** scoring de deux molécules (une molécule requête fixe contre une molécule candidate provenant d'une chimiothèque). a est le seul paramètre de la méthode : il contrôle la longueur de la base du triangle utilisé par la fonction noyau. Cette fonction permet de transformer un ensemble de deltas de Kronecker de même signe en une fonction continue.

chimistes et peut s'exécuter en parallèle.

Initialement, nous voulions créer une méthode sans paramètre. Nous n'avons pas réussi mais obtenu à la place une méthode avec un seul paramètre explicite. La méthode possède des paramètres implicites tels que le mode de calcul des charges partielles ainsi que la façon dont les conformations sont calculées. Une méthode sans paramètre serait peut-être moins efficace. En effet, le paramètre peut être réglé différemment pour chaque espace de recherche.

Dans ACPC-1.2, la méthode d'encodage de la molécule a été modifiée : on n'utilise plus le « linear binning »[170] mais une fonction noyau triangulaire. C'est cette fonction noyau qui introduit un paramètre : la longueur dx de la demi base du triangle (il a pour hauteur un) qui permet de compter une contribution pour deux deltas de Kronecker suffisamment proches. Soit $q_i\delta_{kd_q}$ un delta de la molécule requête. Soit $q_j\delta_{kd_c}$ un delta de la molécule candidate. La contribution au score des deux molécules pour ces deux deltas est $q_iq_j(1.0 - \frac{abs(d_q-d_c)}{dx})$ si $abs(d_q - d_c) < dx$, zero sinon.

La version précédente du logiciel utilisait plus de mémoire. Le vecteur inter-atomique le plus long de la molécule encodée conditionnait la taille du vecteur à conserver en mémoire. Grâce à la fonction noyau, seul le nombre d'atomes de la molécule encodée influence la quantité de mémoire nécessaire. Une molécule avec N atomes est encodée via une liste de $\frac{N(N-1)}{2}$ deltas de Kronecker. On utilise un arbre d'intervalles[33] pour indexer la molécule

requête et ainsi accélérer le calcul du score (figure 4.2).

4.2.1 Nouveau jeu de données

ACPC-1.2 a été validé sur un jeu de données plus difficile et complet que la version précédente : on utilise DUDE au lieu de DUD. Ce jeu de données a été découpé en un jeu d'apprentissage ($\text{DUDE}_{apprentissage}$) et un jeu de test ($\text{DUDE}_{test}$). Le jeu d'apprentissage se compose de 51 protéines choisies aléatoirement (sur un total de 102), avec seulement le premier ligand dans la liste de ligands utilisé comme requête. Le jeu de test : toutes les protéines cibles restantes avec tous leurs ligands. On note qu'il n'y a pas de chevauchement entre le jeu d'apprentissage et le jeu de test, ceci afin d'éviter de biaiser positivement les résultats lors des tests (qui essaient de reproduire des conditions d'utilisation réelles).

DUDE comprend 102 protéines cibles, 25660 ligands et 1060858 molécules supposées inactives. Toutes les molécules ont été préparées en suivant le protocole suivant : i) filtrage par InChi unique via Open Babel[116] 2.3.9 pour enlever les doublons ii) génération avec OMEGA[65] 2.4.6 de la conformation de plus basse énergie pour chaque molécule iii) assignation des charges partielles en utilisant le champ de force MMFF94x de MOE 2013.08 iv) renommage des molécules pour suivre le schéma [active]InChIKey[InChIKey].

4.2.2 Réglage du paramètre pour chaque espace chimique

La méthode ACPC est suffisamment générique pour que toute propriété chimique qui peut être représentée par un nombre centré sur certains atomes puisse être utilisée. ACPC-1.2 peut travailler indifféremment dans l'espace électrostatique (ES), stérique (S) ou hydrophobe (H). ACPC peut aussi travailler dans l'espace combiné des trois.

Sur le jeu d'apprentissage, les scores pour chaque espace de recherche ont été centrés et normalisés. Ensuite, une combinaison optimale des poids de chaque espace a été calculée afin d'augmenter l'AUC moyenne et l'enrichissement à 1%. Les poids optimaux sont de 0.8 pour l'espace électrostatique, 0.2 pour l'espace stérique et 0.5 pour l'espace hydrophobe. Ces poids établissent aussi l'importance relative de chaque espace de recherche : la composante électrostatique est la plus importante, suivie de près par la composante hydrophobe alors que la contribution de la composante forme de la molécule est moindre. Ceci dit,

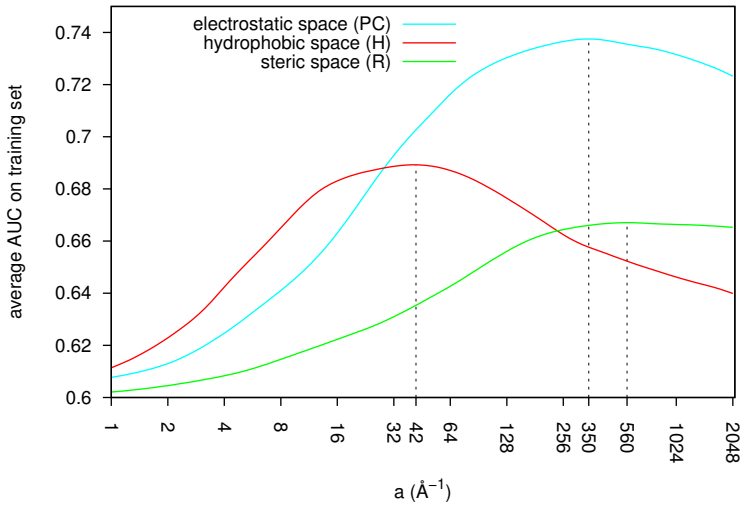


FIGURE 4.3 – En utilisant le jeu d'apprentissage, une valeur optimale du paramètre a été mise au point pour chaque espace de recherche. La valeur optimale est indiquée en pointillé pour chaque espace.

ACPC encode partiellement la forme de la molécule quel que soit l'espace de recherche utilisé puisque la position des atomes (mais sans leur rayon pour les espaces ES et H) est toujours utilisée lors du calcul des distances inter-atomiques.

4.2.3 Performances de la nouvelle version

La performance du logiciel a été évaluée en utilisant l'aire sous la courbe ROC mais aussi l'enrichissement à 1% qui avait été négligé lors de l'évaluation précédente. Sur la figure 4.4, on peut noter deux faits importants : ACPC dans l'espace combiné ES, H et S (ACPCRH) est la méthode la moins aléatoire. En effet, c'est la méthode qui a le moins de chance de donner une aire sous la courbe ROC inférieure à 0.5. C'est aussi la méthode qui atteint les taux d'enrichissement les plus hauts. Un enrichissement de 85 est atteint au moins une fois lors des expériences sur le jeu de test.

4.2.4 Requêtes consensus

Il est possible de faire des requêtes consensus si l'on dispose de plusieurs molécules actives connues. Pour une discussion et deux méthodes de consensus utilisables en LBVS,

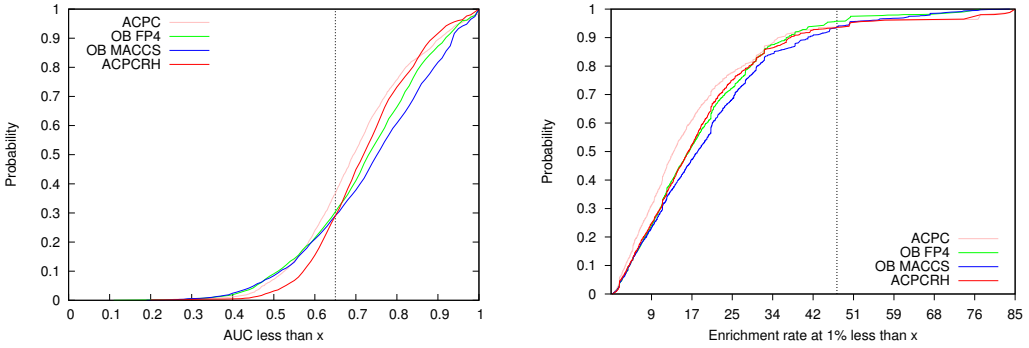


FIGURE 4.4 – **À gauche :** fonction de répartition cumulée pour ACPC dans l'espace électrostatique (ACPC), les signatures FP4 et MACCS d'Open Babel ainsi que pour ACPC utilisant trois espaces en simultané (ACPCRH) sur le jeu de données DUDE-test. **À droite :** enrichissement à 1% lors de la même expérience.

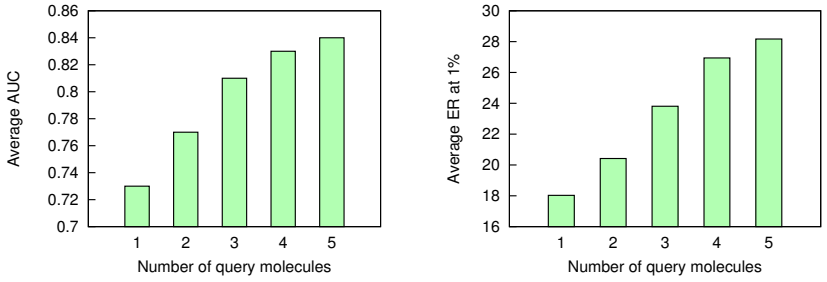


FIGURE 4.5 – **À gauche :** AUC moyenne en fonction du nombre de molécules actives connues utilisées durant une requête consensus. **À droite :** enrichissement à 1% lors de la même expérience.

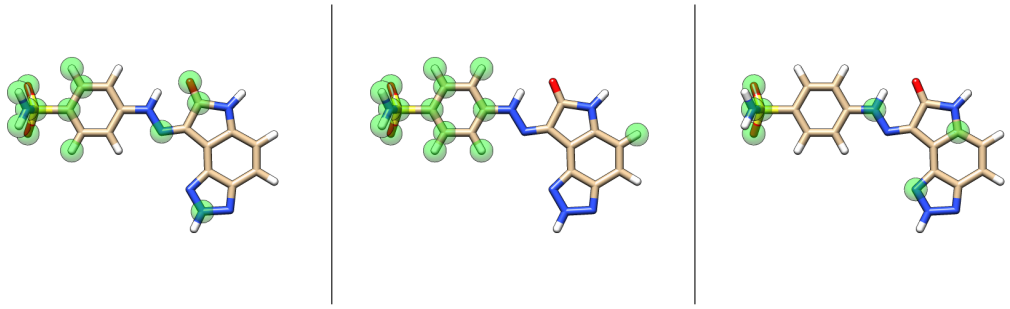


FIGURE 4.6 – Dans chaque espace de recherche, la molécule requête peut être annotée atome par atome en fonction de la contribution à l'AUC. À gauche : exemple dans l'espace électrostatique, au milieu l'espace stérique et à droite l'espace hydrophobe. La molécule montrée est le premier ligand de la cible PDB:cdk2 avec en vert les atomes qui si enlevés feraient décroître l'AUC.

on peut se rapporter à l'article de Hert [66]. Pour une discussion plus générale sur le scoring par consensus, on peut se rapporter à Feher [42]. Feher considère le consensus dans le cas du docking et de la prédiction d'énergie libre de liaison de complexes protéine-ligand mais les méthodes de consensus dont il parle sont aussi applicables en LBVS.

4.2.4.1 Protocole

Chaque molécule requête est utilisée tour à tour afin d'ordonner les molécules de la chimiothèque. Ensuite, le rang minimum est utilisé pour ordonner les molécules. On assigne à une molécule candidate le rang le plus petit qui lui a été assigné par l'une des molécules requête.

4.2.4.2 Évaluation

Pour évaluer les performances de la méthode, cinq molécules requêtes on été choisies aléatoirement dans le jeu de données DUDE-apprentissage. Ensuite, de une à cinq molécules requêtes ont été utilisées simultanément dans une requête consensus dans l'espace électrostatique. L'amélioration ainsi obtenue de l'aire sous la courbe ROC ainsi que de l'enrichissement à 1% est visible sur la figure 4.5.

4.2.5 Annotation de la molécule requête

La molécule requête R peut être annotée automatiquement, dans un espace donné, afin de voir quels sont les atomes importants pour trouver d'autres molécules actives. La méthode suivante, à l'intention des chimistes, permet d'annoter une molécule si l'on dispose d'une chimiothèque contenant plusieurs molécules actives connues :

1. On utilise r telle quelle et on mémorise l'ASC obtenue que l'on note ASC_r (aire sous la courbe ROC de référence).
2. On considère toutes les variantes de R qui consistent à ignorer un de ses atomes. Si R a N atomes, on obtient N variantes. Chaque variante est une molécule virtuelle (notée V_i), dans laquelle l'atome (d'indice i) est ignoré lors de l'encodage.
3. On fait ensuite une requête avec chaque variante V_i de R. On note ASC_i l'aire sous la courbe ROC ainsi obtenue. Si $\mathrm{ASC}_i < \mathrm{ASC}_r$ alors on conclue que l'atome i de R est important dans la molécule requête (figure 4.6 ; atomes dans une boule verte transparente).

On remarque que cette méthode est générique. En effet, elle peut être utilisée pour tout espace de recherche supporté par ACPC (figure 4.6). Nous pensons que cette fonctionnalité permet de souligner un motif structural qui serait partagé par plusieurs molécules actives. Dans la figure 4.6, il semble que la partie gauche de la molécule comporte toujours plusieurs atomes soulignés, quel que soit l'espace de recherche considéré.

4.2.6 Exécution parallèle

Le logiciel ACPC étant implémenté dans un langage fonctionnel souvent utilisé de façon pure (c'est à dire via des fonctions sans effet de bord), sa parallélisation est triviale et permet d'atteindre une fréquence de traitement des molécules encore plus élevée sur une machine multi-cœurs. Une fois de plus, nous avons utilisé la librairie Parmap [29] à cet effet.

Les performances de la version parallèle d'ACPC ont été mesurées et comparées avec la performance des signatures MACCS et FP4 telles qu'implémentées dans la librairie Open Babel qui est programmée en C++. Les résultats sont visibles dans la figure 4.7. Chaque

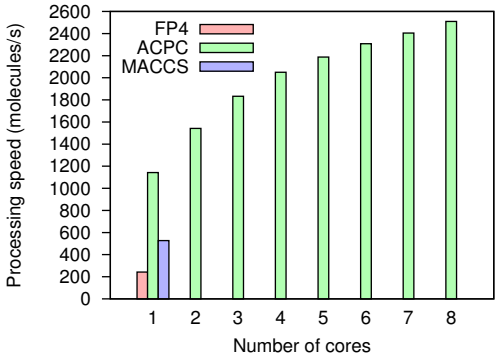


FIGURE 4.7 – Comparaison de vitesse entre ACPC dans l'espace électrostatique et Open Babel 2.3.9 utilisant la signature MACCS ou FP4. La cible est la protéine PDB:hivpr qui comporte un total de 26450 ligands et leurres.

méthode à été lancée trois fois, après une première exécution non chronométrée. On a rapporté sur la figure la moyenne des trois exécutions chronométrées.

Chapitre 5

Discussion

Dans ce chapitre, nous discutons les résultats présentés dans les parties précédentes. Nous donnons aussi quelques pistes afin d'aller plus loin avec certaines idées que nous avons explorées. Nous mettons aussi en avant des points particulièrement intéressants et des limitations de certains travaux.

5.1 Le calcul parallèle et distribué

Le logiciel PAR a été utilisé afin d'obtenir rapidement bon nombre des résultats qui sont présentés dans cette thèse. La plupart du temps, ces résultats sont présentés sous forme de figure et certaines figures ont nécessité des dizaines de milliers d'expériences computationnelles afin d'obtenir tous les points d'une courbe. Nous utilisons encore PAR, ainsi que plusieurs de nos anciens collègues. PAR peut être utile à de nombreux scientifiques. PAR nous a permis plusieurs fois d'éviter d'utiliser le supercalculateur (surchargé d'utilisateurs) de notre campus.

On peut cependant noter trois points qui font défaut dans PAR. Tout d'abord, PAR ne gère pas les données hormis les options passées aux programmes. Il vaudrait mieux utiliser PAR sur des machines disposant d'un système de fichier distribué [54,124,129,131] pour exécuter des expériences intensives en données. Ensuite, les communications réseau de PAR ne sont pas sécurisées. C'est problématique dans un contexte de recherche scientifique où certains résultats sont destinés à être brevetés. On peut citer au moins GNU parallel [150], comme compétiteur à PAR sécurisant les communications. Enfin, PAR n'est pas tolérant

aux fautes. Cela limite son utilisation aux expériences de taille petite a moyenne (i.e. utilisant moins de 200 ordinateurs pendant quelques heures). Pour les expériences nécessitant des calculs à plus grande échelle, on peut recommander CONDOR[100].

5.2 La comparaison électrostatique

EleKit est à ce jour et à notre connaissance le seul outil qui permet de comparer une protéine ligante avec une petite molécule dans l'espace électrostatique. En général, on observe bien une corrélation entre les molécules actives et la protéine ligante.

5.2.1 Parallélisation quasi optimale

Le logiciel EleKit est assez lent dans sa version séquentielle. Non pas parce que le logiciel est inefficace mais parce que pour chaque molécule que l'on veut comparer, il faut faire un calcul explicite du champ électrostatique autour de cette molécule, calcul qui est assuré par le solveur APBS. Pour chaque petite molécule et en utilisant un seul cœur, APBS met entre une à deux minutes pour faire ce calcul. Heureusement, EleKit permet de traiter en parallèle des molécules différentes sur une machine multi-cœurs.

```
   let completed = open_out out_files     in
   (* molecules from a multiple molecules mol2
      file are processed in parallel *)
-  L.iter
+  Parmap.pariter ~ncores:nprocs
     (fun (pdb_B', m_name) ->
        P.printf "creating .pqr for %s...\n%!" pdb_B';
        let maybe_pqr_B =
@@ -170,7 +170,7 @@ let main () =
          (* %! --> flush out *)
          P.fprintf completed
            "%s %s %s\n%!" dx_out_B_gz mask_out_B m_name)
-    molecules;
+    (Parmap.L molecules);
   close_out completed
 ;;
```

FIGURE 5.1 – Modification pour paralléliser le logiciel EleKit. Deux lignes modifiées changent un algorithme séquentiel en un algorithme parallèle.

Nous avons utilisé la librairie Parmap [29] pour paralléliser le code d'EleKit. EleKit est un logiciel de 2734 lignes de code OCaml (un langage de programmation assez concis). Grâce à la librairie Parmap, rendre le logiciel EleKit parallèle ne nécessite que le changement de deux lignes du programme (figure 5.1). La majeure partie du développement et du débogage du logiciel à été faite sur du code séquentiel, ce qui est très confortable. Par

contre, les grandes expériences nécessaires à la validation scientifique du logiciel ont été réalisées en utilisant la version parallèle.

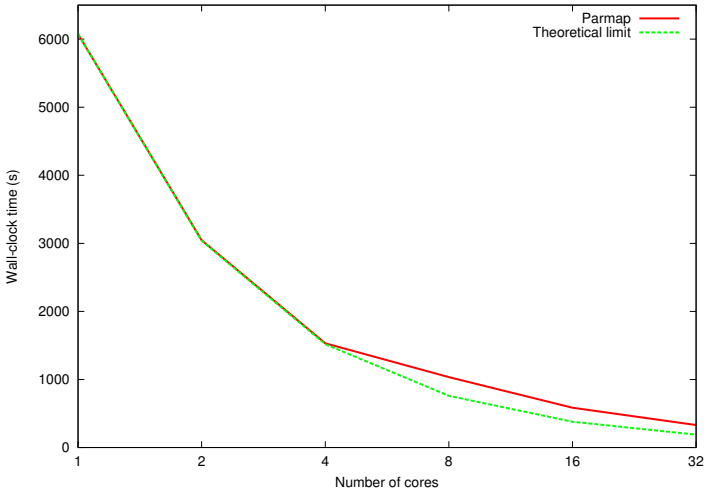


FIGURE 5.2 – Temps d'exécution d'EleKit en utilisant de un à 32 cœurs sur un noeud de supercalculateur.

Malgré le peu d'effort nécessaire à la parallélisation du logiciel, les performances sont excellentes (figure 5.2). Jusqu'à quatre cœurs, la performance mesurée est quasiment indistinguable de la limite $\frac{\text{temps\_sequentiel}}{\text{nombre\_de\_processeurs}}$. De plus, cette méthode est largement applicable en chemoinformatique où l'on traite souvent un grand nombre de molécules indépendantes les unes des autres. Il suffit que la boucle principale du programme utilise un tableau ou une liste avec, par exemple, une molécule à traiter dans chaque case du tableau ou élément de la liste. Ensuite, la librairie Parmap peut être utilisée pour transformer trivialement ce programme séquentiel en un programme parallèle. Avec Parmap, les performances sont bonnes si le calcul pour chaque molécule a une granularité moyenne à grosse. Par exemple, s'il faut au moins une seconde pour traiter chaque molécule. Dans un tel cas d'utilisation, nous ne pouvons que recommander Parmap pour du calcul scientifique.

5.2.2 Applicabilité et limitations

Le logiciel EleKit permet bien de calculer une similarité dans l'espace électrostatique. Par contre, nous ne fournissons pas de règles quant à l'applicabilité du logiciel. Par exemple,

étant donné un complexe protéine-protéine, est-ce qu'il est judicieux d'utiliser EleKit pour scorer les molécules destinées à perturber cette interface protéine-protéine. Peut-être que quantifier la polarité d'une interface protéine-protéine ainsi que déterminer un seuil permettrait de définir une telle applicabilité.

Pour interpréter les résultats d'EleKit, nous considérons qu'un score de corrélation inférieur à 0,2 indique une mauvaise similarité électrostatique de la molécule considérée avec la protéine qu'elle est censée remplacer.

On peut aussi remarquer d'autres limites de son applicabilité : si les ligands sont mal positionnés lors de l'étape de docking, alors le score calculé par EleKit est faux. Si le ligand subit un ajustement induit après positionnement dans le site actif (et donc un changement de conformation), le score calculé par EleKit est là aussi faux...

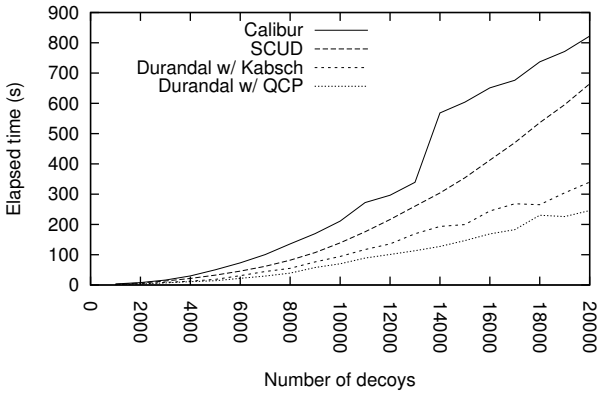


FIGURE 5.3 – Comparaison de vitesse entre Calibur, SCUD (une méthode approximative), Durandal et Durandal-QCP sur des modèles de la protéine PDB:1SHFA avec une distance seuil de 1.5Å. Pour simplifier, seule une tranche du paysage 3D obtenu en variant le seuil de 0.5 à 8Å par pas de 0.5Å est montrée. Durandal est plus rapide que Calibur dans 98% de la totalité de ce paysage. Les résultats pour SCUD et Durandal-QCP ont été calculés seulement pour cette tranche.

5.3 L'algorithme de regroupement exact

À ce jour, nous ne connaissons pas d'implémentation de l'algorithme de regroupement exact plus efficace que celle implémentée dans Durandal. Par contre, il existe plusieurs logiciels qui se disent plus rapides mais qui implémentent en fait soit un autre algorithme de regroupement soit introduisent des approximations dans l'algorithme ou dans la mesure de distance entre les modèles[62,74,183,187].

Depuis sa publication, Durandal a été utilisé dans des comparaisons de méthodes de clustering[30,74,187]. Il a aussi été utilisé pour sélectionner des modèles à la suite de simulations de pliage de protéines[11,103,134,138,139,140,143,144,148].

5.3.1 Nouvelle méthode de calcul du RMSD

Par la suite, le logiciel Durandal a encore été accéléré (figure 5.3) grâce à une méthode récente de calcul du RMSD[154]. Son architecture logicielle à été décrite[14] afin de permettre éventuellement à d'autres chercheurs de modifier le logiciel. Le seul point négatif de Durandal est sa consommation mémoire. Durandal ne peut faire le calcul du regroupement que si la moitié de la matrice de dissimilarité (diagonale exclue) tient en

mémoire.

5.3.2 Choix de la distance seuil

Alors que Durandal essaie de faire le calcul du regroupement le plus vite possible, il ne dit rien quant à la façon de choisir la distance seuil à utiliser lors de ce calcul. Une méthode intéressante nous à été suggérée par le docteur Dominik Gront dans une communication privée. On a besoin de faire le regroupement plusieurs fois puisqu'on veut éviter deux extrêmes. Le premier est celui où l'on n'a qu'une seule partition avec toutes les protéines incluses. Le deuxième est celui où l'on a un groupe par protéine. L'idéal est donc de se positionner entre ces deux extrêmes. On a donc besoin de faire le calcul du regroupement plusieurs fois. Il existe aussi une statistique appelée « gap » qui permet d'estimer de façon automatique le nombre de partitions d'un jeu de données[156]. Par rapport à d'autres méthodes, cette statistique est intéressante puisqu'elle supporte aussi le cas où il n'y a qu'une seule partition.

5.3.3 Choix des références

Concernant les logiciels Durandal et Fragger, il existe peut-être une amélioration possible en ce qui concerne le choix des références. Une référence est un point dont on se sert pour créer un index. C'est à dire que l'on mesurera la distance de ce point à tous les autres. Actuellement, la manière utilisée est de tirer les références aléatoirement dans Durandal et de laisser l'utilisateur les choisir dans Fragger. Cependant, dans la littérature on trouve une indication sur les propriétés que doit satisfaire un point qui sert de référence pour créer un index. Les points de références doivent être loin des centres de groupes[135]. Au cours des expériences, nous avons observé que Durandal utilise seulement deux ou trois références pour accélérer l'initialisation de la matrice de dissimilarité. Implémenter une heuristique permettant de choisir judicieusement les points de références serait une amélioration possible.

5.4 Autocorrélation des charges partielles

5.4.1 Amélioration de la diversité chimique

Si l'on considère qu'ACPC est une méthode qui ne promeut pas assez la diversité des molécules, on peut considérer deux éventualités pour palier à ce défaut.

La méthode la plus radicale consisterait à travailler dans l'espace des points pharmacophoriques au lieu des espaces ES, H ou S. Le problème consiste alors à calculer la fonction d'autocorrélation dans un tel espace.

Enfin, pour les utilisateurs éventuels qui souhaiteraient atteindre des taux d'enrichissement encore plus hauts même au détriment de la diversité chimique, on peut citer la méthode fmcsR[171]. fmcsR est une méthode de MCS tolérante aux erreurs, qui permet à l'utilisateur de fixer un nombre maximal d'erreurs tolérées (en terme de discordance de type d'atome ou de liaison chimique) lors de la recherche de MCS. La librairie ChemmineR pour le logiciel de statistiques R est l'implémentation libre de référence de cette méthode. Mais, le logiciel est très lent (compter plusieurs secondes pour traiter une seule molécule) et les molécules avec un trop grand nombre d'atomes sont ignorées par le logiciel afin de ne pas faire exploser le temps de calcul.

5.4.2 Autres utilisations possibles du descripteur

Nous ne sommes pas défenseur de la société « big brother » dans laquelle les gens sont constamment filmés, enregistrés et pistés. Cependant, nous ne pouvons nous empêcher de remarquer que la méthode ACPC peut sans doute être étendue à d'autres domaines : par exemple à la reconnaissance d'empreintes digitales ou à la reconnaissance de visages en 3D.

Il faut tout de même garder à l'esprit qu'ACPC n'est pas invariant au changement d'échelle ou à une quelconque distorsion de l'ensemble de points. Si les empreintes digitales sont toutes photographiées à la même échelle et exemptes de distorsions (ou ont subies une correction de distorsion préalable), une méthode telle qu'ACPC devrait marcher pour encoder et reconnaître des empreintes digitales. Pour faire de la reconnaissance de visages, il faudrait des points caractéristiques issus de photographies 3D de visages tels

que les extrémités de la bouche, du nez et des yeux. Dans l'idéal, le sujet pris en photo tridimensionnelle ne devrait pas faire de grimace car cela introduirait une distorsion.

Au passage, notons qu'ACPC à été paramétré pour donner de bons résultats sur de petites molécules. Il n'est donc pas conseillé d'utiliser le logiciel tel quel pour comparer, par exemple, des protéines. Pour comparer seulement les sites actifs de ces protéines, la méthode est probablement applicable.

5.5 Sélection de fragments de protéines

Le logiciel Fragger est utilisé par nos collègues faisant de la conception de protéines afin de créer des librairies de fragments.

5.5.1 Aller plus vite

Citons ici deux pistes qui devraient permettre d'accélérer encore plus une requête structurale. La méthode la plus facile à mettre en place consiste peut-être à considérer une distance entre fragments extrêmement rapide à calculer telle que celle utilisée dans le logiciel BCSearch [59,60].

Une méthode plus compliquée à mettre en place consiste à précalculer à l'avance un index pour tous les fragments d'une longueur donnée. Par exemple, en calculant un "Geometric Near-Neighbor Access Tree" [17]. Malheureusement, le calcul d'un tel index à l'échelle de la PDB est coûteux et devrait être fait à l'avance pour chaque taille de fragment requête susceptible d'être utilisée.

Chapitre 6

Conclusion

Nous récapitulons ici les contributions fournies durant cette thèse avant de lister quelques recommandations qui pourraient être utiles à un jeune thésard en bioinformatique structurale ou chémoinformatique.

Notre première contribution consiste en un logiciel libre (PAR) qui permet d'exécuter de façon parallèle et distribuée des commandes UNIX indépendantes. Cet outil a été utilisé maintes fois dans les travaux mentionnés ici afin d'accélérer les expériences qui ont permis de valider scientifiquement et de mesurer la performance des logiciels présentés. PAR est encore utilisé par au moins deux de mes anciens collègues qui apprécient sa simplicité d'utilisation et l'accélération notable qu'il permet d'atteindre lors de campagnes d'expériences computationnelles. Nous pensons que PAR pourrait être utile à de nombreux autres scientifiques.

Ensuite, nous avons présenté un algorithme qui permet d'accélérer de façon significative le calcul du regroupement exact pour des modèles de protéines. Le regroupement exact est largement utilisé en protein folding et lors de l'analyse de traces de simulations de dynamique moléculaire. Avec notre implémentation (le logiciel Durandal) nous avons montré que les performances de notre méthode sont supérieures à celles d'une méthode approximée. Durandal permet donc de faire du regroupement en considérant un plus grand nombre de modèles que précédemment, ou plus rapidement si l'on ne souhaite pas considérer plus de modèles.

Nous avons aussi créé un algorithme permettant de comparer dans l'espace électro-

statique une protéine avec la petite molécule destinée à la remplacer. Cet algorithme est implémenté dans le logiciel EleKit et a été utilisé lors de la compétition SAMPL4 afin d'améliorer la performance d'un entonnoir de criblage virtuel [167]. Pour la petite histoire, l'équipe ayant utilisé EleKit à gagné la compétition. Mais surtout, EleKit est un outil de mesure pour les gens qui essaient de perturber une interface protéine-protéine. EleKit permet de vérifier qu'une molécule candidate est similaire dans l'espace électrostatique avec la protéine qu'elle est censée remplacer. Si le score de corrélation calculé par EleKit est inférieur à +0.2, c'est un mauvais signe pour la molécule considérée.

Enfin, probablement notre contribution la plus importante, nous avons proposé une technique combinant la fonction d'autocorrélation avec une fonction noyau triangulaire afin d'encoder finement une molécule 3D dans tout espace chimique qui peut être décrit par des valeurs atome-centrées. Cette méthode, implémentée dans le logiciel ACPC, montre d'excellents résultats en terme d'aire sous la courbe ROC et d'enrichissement lors d'expériences à posteriori de criblage virtuel. La vitesse de la méthode est remarquable, avec plus de 1600 molécules traitées par secondes sur un ordinateur standard et la possibilité d'aller encore plus vite sur une machine multi-cœur (plus de 2400 molécules par seconde avec huit processeurs). Les différentes versions du logiciel ACPC ont aussi donné lieu à la mise à disposition de la communauté scientifiques de versions préparées pour les expériences en LBVS des jeux de données DUD et DUDE. Préparer un jeu de données est une tâche fastidieuse mais qu'il faut mener avec soin afin de ne pas introduire d'erreurs ou d'omissions dans le jeu de données. Nous espérons que ces jeux de données permettront d'accélérer les travaux d'autres chercheurs du domaine et faciliteront la comparaison des résultats entre différentes méthodes.

6.1 Recommandations à un jeune doctorant

Pour finir, nous souhaitons mentionner quelques recommandations pour quelqu'un qui commencerait une thèse en bioinformatique ou chemoinformatique. Ces recommandations sont librement inspirées des recommandations de Bosco K. Ho [1]. À l'origine, ces recommandations sont à destination des biologistes computationnels mais elles ont en fait une

1. `http://boscoh.com/protein/notes-to-a-young-computational-biologist.html`

portée bien plus grande. L'auteur original de ces recommandations est un chercheur en dynamique moléculaire et semble être un chercheur productif de son domaine, avec une grande expérience du développement logiciel pour la recherche. Le docteur Ho est spécialiste des expériences computationnelles longues, avec de nombreux paramètres et utilisant des logiciels complexes. Nous pensons que ces recommandations sont utiles car elles reflètent les règles que l'on peut acquérir après des années de pratique. Certaines de ces règles sont acquises après des expériences malencontreuses, comme la perte de données expérimentales durement acquises. Certaines de ces recommandations sont à destination de tout programmeur et sont en fait issues du livre « The pragmatic programmer »[155].

Être familier avec l'environnement Linux et en particulier la ligne de commande[147]. Avec des outils tels que ssh, GNU parallel ou PAR : utiliser un ou 128 ordinateurs n'est pas si différent. Par contre, il y a une différence de productivité certaine.

Maîtriser un langage de script. Par exemple : Python, Ruby, zsh, awk ou bash, pour les tâches que l'on veut faire vite mais pas forcément proprement. Par exemple, des logiciels très courts (dix lignes au plus) à usage unique pour préparer une expérience ou mettre en forme des résultats. Python est suffisamment simple pour être recommandé à tout scientifique qui devrait se mettre à la programmation[106].

Apprendre à utiliser R ou la librairie numpy[7] de Python pour faire des statistiques sur ses données[75]. Pour une utilisation scientifique, on rencontre vite les limites d'un tableur, surtout si on a beaucoup de données. J'ai vu un collègue développer des logiciels de bioinformatique structurale avec R à une vitesse impressionnante. Il existe des librairies R pour presque tous les domaines scientifiques.

Sauver le plus de choses possibles dans des fichiers texte[121]. On peut toujours lire un fichier texte, ce n'est pas vrai des formats binaires qui peuvent nécessiter une version particulière d'un certain logiciel. Aussi, les fichiers textes sont faciles à mettre sous contrôle de version. Ils peuvent aussi être formatés de façon à être faciles à lire et à écrire par un programme. Même si l'on ne travaille pas dans un laboratoire réel avec une blouse blanche, un scientifique doit garder une trace de ce qu'il a fait. Ces données seront utiles lors d'une éventuelle publication, afin de décrire les conditions des expériences qui ont été menées.

Tout mettre sous contrôle de version. Le code source des logiciels bien-sûr mais aussi les fichiers de configuration et les scripts des expériences ainsi que les jeux de données et les résultats finaux. Quand on prépare un jeu de données, il est bien utile de pouvoir vérifier les modifications successives et de pouvoir en annuler une si besoin. Un outil de différence graphique (tel que tkdiff) permet de visualiser la différence entre deux versions successives. Une telle précaution avant d'enregistrer une nouvelle version évite de nombreuses erreurs qui peuvent être difficiles à investiguer sinon.

Organiser ses jeux de données dans une arborescence avec une structure régulière. Cela facilite l'écriture des scripts des expériences et peut même autoriser certaines requêtes sur le jeu de données sans avoir besoin de l'importer dans une base de données. Chaque niveau de l'arborescence doit avoir du sens. Par exemple, l'arborescence dans notre version préparée pour les expériences en LBVS du jeu de données DUDE est la suivante : protéine_cible/ligand_N/N.mol2|pqr|pl. Chaque répertoire protéine_cible contient un fichier ligdecs.1conf.mol2|pqr|pl qui signifie tous les ligands et tous les leurres dans leur conformation de plus basse énergie. Chaque protéine cible est isolée dans son propre répertoire. Chaque ligand est aussi isolé afin de pouvoir être utilisé comme requête. Les fichiers 'mol2' sont pour l'espace électrostatique, les 'pqr' pour l'espace stérique et les 'pl' pour l'espace hydrophobe.

Apprendre un programme en ligne de commande pour faire des graphes et des courbes. R permet cela. Les utilisateurs de Python pourront utiliser MatPlotlib tandis que les autres pourront utiliser gnuplot. La majorité des figures de cette thèse sont faites avec gnuplot ou xfig. Un bon article contient des images. Ce n'est pas tout d'avoir obtenu des données expérimentales, encore faut-il les montrer de façon claire afin que les lecteurs voient aussi ce que vous essayez de montrer. Un bon graphique montre une seule chose à la fois. Ce n'est vraiment que si l'on est contraint par la place que l'on doit créer des graphiques plus compliquées qui combinent plusieurs informations. Cela sera souvent au détriment de la lisibilité du graphique. Pour gnuplot, le livre « Gnuplot in action »[76] du physicien Philipp K. Janert ainsi que le site « Gnuplot not so FAQ »[2] du docteur Toshihiko Kawano du LANL (physicien lui aussi) sont des ressources inestimables. De même, on préférera

2. `http://folk.uio.no/hpl/scripting/doc/gnuplot/Kawano/index-e.html`

une figure à un tableau. Un tableau est peut-être facile à traiter par un ordinateur mais les humains préfèrent de l'information visuelle. Une tendance peut être difficile à voir dans un tableau mais elle sautera aux yeux avec une courbe.

Maîtriser un éditeur de texte avancé. On peut faire beaucoup de mise en forme de données de façon interactive dans un tel éditeur de texte. On est aussi très productif lorsque l'on programme ou que l'on rédige un texte quelconque. Pour les même raisons de productivité, il est utile de savoir taper au clavier sans le regarder.

Il faut tout automatiser. On est souvent obligé de relancer une expérience après un changement de paramètre ou une modification du logiciel. Une bonne expérience est une expérience qui se lance de façon triviale (par exemple via l'exécution d'un seul script) et qui va jusqu'à mettre en forme les résultats pour la publication.

Choisir avec soin le niveau d'abstraction dans lequel on souhaite travailler et faire de la recherche. Quand on écrit un programme soi-même, c'est afin d'avoir le contrôle total et la compréhension profonde de ce que l'on fait. Les bons chercheurs et les sportifs de haut niveau ont le souci du détail. Mais le temps est précieux et on ne peut pas tout programmer soi-même. On n'est pas non plus expert dans tous les domaines. Pour tous les niveaux d'abstraction inférieurs au niveau où l'on souhaite faire de la recherche : ne pas réinventer la roue mais utiliser des programmes et librairies existants.

Quand on modifie un grand logiciel (appelons le GL) fait par quelqu'un d'autre : utiliser la « technique de la seringue ». À moins que l'on ne fasse partie de l'équipe qui développe GL, il est très dur de faire accepter ses changements dans la version officielle. Et, à cause des prochaines évolutions de GL, vos modifications ne seront probablement plus compatibles. La « technique de la seringue » consiste à extraire de GL juste les données dont on a besoin, faire ses calculs dans son propre logiciel, puis injecter les résultats au bon endroit. Ainsi, on dépend moins du code de GL et on travaille librement sur son propre projet, qui peut être programmé dans un autre langage. Au besoin, seules les parties extraction des données et injection des résultats (notre seringue virtuelle) devront être maintenues.

Enfin, pour faire des prototypes de logiciel pour la recherche, nous recommandons d'utiliser des langages du plus haut niveau possible. En effet, si le but de la recherche est

de montrer qu'une idée marche, il faut pouvoir tester des idées de préférence vite mais correctement. Hors, plus on utilise un langage de haut niveau, plus on est productif. Mon parcours a été d'utiliser Python, puis C++ pour finir avec OCaml. Je recommanderais plutôt Python pour les non informaticiens et les logiciels très petits. Pour le reste, des langages tels qu'OCaml[94] ou Haskell[72] permettent de détecter à la compilation de nombreuses erreurs. L'inférence de type est une autre raison qui fait qu'on programme rapidement avec ces langages : le type des variables est déduit mathématiquement par le compilateur et libère donc le programmeur de leur écriture. En utilisant un langage de haut niveau, on passe moins de temps à programmer (et à déboguer) et on peut donc réfléchir plus à son sujet de recherche.

Chapitre 7

Bibliographie

Bibliographie

[1] Adams, P. D., Afonine, P. V., Bunkóczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L.-W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C., and Zwart, P. H. (2010). *PHENIX* : a comprehensive Python-based system for macromolecular structure solution. *Acta Crystallogr. Sect. D*, **66**(2), 213–221.

[2] Allouche, D., André, I., Barbe, S., Davies, J., de Givry, S., Katsirelos, G., O'Sullivan, B., Prestwich, S., Schiex, T., and Traoré, S. (2014). Computational protein design as an optimization problem. *Artificial Intelligence*, **212**, 59 – 79.

[3] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM.

[4] Anderson, D. P. (2004). BOINC : A system for public-resource computing and storage. In R. Buyya, editor, *Proceedings of the 5th International Workshop on Grid Computing (GRID'04)*, pages 4–10, Pittsburgh, PA, USA. IEEE Computer Society.

[5] Anfinsen, C. B. (1973). Principles that govern the folding of protein chains. *Science*, **181**(96), 223–230.

[6] Anzali, S., Barnickel, G., Krug, M., Sadowski, J., Wagener, M., Gasteiger, J., and Polanski, J. (1996). The comparison of geometric and electronic properties of molecular surfaces by neural networks : Application to the analysis of corticosteroid-binding globulin activity of steroids. *Journal of Computer-Aided Molecular Design*, **10**(6), 521–534.

[7] Ascher, D., Dubois, P. F., Hinsen, K., Hugunin, J., Oliphant, T., *et al.* (2001). Numerical Python.

[8] Baker, N. A., Sept, D., Joseph, S., Holst, M. J., and McCammon, J. A. (2001). Electrostatics of nanosystems : application to microtubules and the ribosome. *Proc Natl Acad Sci U S A.*, **98**(18), 10037–41.

[9] Barbosa, F. and Horvath, D. (2004). Molecular similarity and property similarity. *Current topics in medicinal chemistry*, **4**(6), 589–600.

[10] Bauknecht, H., Zell, A., Bayer, H., Levi, P., Wagener, M., Sadowski, J., and Gasteiger, J. (1996). Locating biologically active compounds in medium-sized heterogeneous datasets by topological autocorrelation vectors : Dopamine and benzodiazepine agonists. *Journal of Chemical Information and Computer Sciences*, **36**(6), 1205–1213.

[11] Belsom, A., Schneider, M., Fischer, L., Brock, O., and Rappsilber, J. (2015). Serum albumin domain structures in human blood serum by mass spectrometry and computational biology. *Molecular and Cellular Proteomics*.

[12] Bender, A. and Glen, R. C. (2004). Molecular similarity : a key technique in molecular informatics. *Org. Biomol. Chem.*, **2**, 3204–3218.

[13] Bender, A., Jenkins, J. L., Scheiber, J., Sukuru, S. C. K., Glick, M., and Davies, J. W. (2009). How Similar Are Similarity Searching Methods ? A Principal Component Analysis of Molecular Descriptor Space. *Journal of Chemical Information and Modeling*, **49**(1), 108–119.

[14] Berenger, F. *et al.* (2012). Durandal : Fast exact clustering of protein decoys. *J. Comput. Chem.*, **33**(4), 471–474.

[15] Berenger, F., Voet, A., Lee, X., and Zhang, K. (2014). A rotation-translation invariant molecular descriptor of partial charges and its use in ligand-based virtual screening. *Journal of Cheminformatics*, **6**(1), 23.

[16] Berman, H. M. *et al.* (2013). The future of the protein data bank. *Biopolymers*, **99**(3), 218–222.

[17] Brin, S. (1995). Near neighbor search in large metric spaces. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.*, pages 574–584.

[18] Broto, P., Moreau, G., and C., V. (1984a). Molecular structures : perception, autocorrelation descriptor and SAR studies. Autocorrelation descriptor. *Eur. J. Med. Chem.*, **19**(1), 66 – 70.

[19] Broto, P., Moreau, G., and C., V. (1984b). Molecular structures : perception, autocorrelation descriptor and SAR studies. Use of the autocorrelation descriptors in the QSAR study of two non-narcotic analgesic series. *Eur. J. Med. Chem.*, **19**(1), 79 – 84.

[20] Brown, N. (2009). Chemoinformatics - an introduction for computer scientists. *ACM Computing Surveys (CSUR)*, **41**(2), 8.

[21] Budowski-Tal, I., Nov, Y., and Kolodny, R. (2010). FragBag, an accurate representation of protein structure, retrieves structural neighbors from the entire PDB quickly and accurately. *PNAS*, **107**(8), 3481–3486.

[22] Carhart, R. E., Smith, D. H., and Venkataraghavan, R. (1985). Atom pairs as molecular features in structure-activity studies : definition and applications. *Journal of Chemical Information and Computer Sciences*, **25**(2), 64–73.

[23] Chen, J. and Martinez, T. J. (2007). QTPIE : Charge transfer with polarization current equalization. A fluctuating charge model with correct asymptotics. *Chemical Physics Letters*, **438**(4), 315 – 320.

[24] Claessens, M., van Cutsem, E., Lasters, I., and Wodak, S. (1989). Modelling the polypeptide backbone with spare parts from known protein structures. *Protein Engineering*, **2**(5), 335–345.

[25] Collier, J. H. *et al.* (2012). Super : a web server to rapidly screen superposable oligopeptide fragments from the protein data bank. *Nucleic Acids Research*, **40**, W334 – W339.

[26] Consortium, U. *et al.* (2008). The universal protein resource (uniprot). *Nucleic acids research*, **36**(suppl 1), D190–D195.

[27] Cornuéjols, A. and Miclet, L. (2010). *Apprentissage artificiel : concepts et algorithmes. Deuxième édition.* Eyrolles.

[28] Dahiyat, B. I. and Mayo, S. L. (1997). De novo protein design : Fully automated sequence selection. *Science*, **278**(5335), 82–87.

[29] Danelutto, M. and Di Cosmo, R. (2012). A "Minimal Disruption" Skeleton Experiment : Seamless Map and Reduce Embedding in OCaml. *Procedia Computer Science*, **9**(0), 1837 – 1846. Proceedings of the International Conference on Computational Science, ICCS 2012.

[30] Dang, H.-V., Schmidt, B., Hildebrandt, A., Tran, T. T., and Hildebrandt, A. K. (2015). Cuda-enabled hierarchical ward clustering of protein structures based on the nearest neighbour chain algorithm. *International Journal of High Performance Computing Applications*.

[31] Das, R., Qian, B., Raman, S., Vernon, R., Thompson, J., Bradley, P., Khare, S., Tyka, M. D., Bhat, D., Chivian, D., *et al.* (2007). Structure prediction for casp7 targets using extensive all-atom refinement with rosetta@ home. *Proteins : Structure, Function, and Bioinformatics*, **69**(S8), 118–128.

[32] Davis, M. E. and McCammon, J. A. (1990). Electrostatics in biomolecular structure and dynamics. *Chemical Reviews*, **90**(3), 509–521.

[33] de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2008). *Computational Geometry : Algorithms and Applications*. Springer, third edition edition.

[34] Desmet, J., Maeyer, M. D., Hazes, B., and Lasters, I. (1992). The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, **356**(6369), 539–542.

[35] Diamond, R. (1988). A note on the rotational superposition problem. *Acta Crystallographica Section A*, **44**(2), 211–216.

[36] Dittmar, P. G., Stobaugh, R. E., and Watson, C. E. (1976). The chemical abstracts service chemical registry system. i. general design. *Journal of Chemical Information and Computer Sciences*, **16**(2), 111–121.

[37] Dolinsky, T. J., Nielsen, J. E., McCammon, J. A., and Baker, N. A. (2004). PDB2PQR : an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. *Nucleic Acids Res.*, **32**(Web Server issue), W665–7.

[38] Eckert, H. and Bajorath, J. (2007). Molecular similarity analysis in virtual screening : foundations, limitations and novel approaches. *Drug Discovery Today*, **12**(5), 225–233.

[39] Engel, T. (2006). Basic overview of chemoinformatics. *Journal of chemical information and modeling*, **46**(6), 2267–2277.

[40] Fechner, U., Franke, L., Renner, S., Schneider, P., and Schneider, G. (2003). Comparison of correlation vector methods for ligand-based similarity searching. *Journal of Computer-Aided Molecular Design*, **17**(10), 687–698.

[41] Fedak, G., Germain, C., Néri, V., and Cappello, F. (2001). Xtremweb : A generic global computing system. In *CCGRID'01*, pages 582–587. IEEE Computer Society.

[42] Feher, M. (2006). Consensus scoring for protein-ligand interactions. *Drug discovery today*, **11**(9), 421–428.

[43] Flicek, P., Amode, M. R., Barrell, D., Beal, K., Billis, K., Brent, S., Carvalho-Silva, D., Clapham, P., Coates, G., Fitzgerald, S., *et al.* (2014). Ensembl 2014. *Nucleic acids research*, **42**(D1), D749–D755.

[44] Forum, M. P. I. (1994). MPI : A message-passing interface standard. Technical Report UT-CS-94-230, Department of Computer Science, University of Tennessee. Tue, 22 May 101 17 :44 :55 GMT.

[45] Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis and density estimation. *Journal of the American Statistical Association*, **97**, 611–631.

[46] Fraley, C., Raftery, A. E., Murphy, T. B., and Scrucca, L. (2012). *mclust Version 4 for R : Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation*.

[47] Gainza, P., Roberts, K. E., Georgiev, I., Lilien, R. H., Keedy, D. A., Chen, C.-Y., Reza, F., Anderson, A. C., Richardson, D. C., Richardson, J. S., *et al.* (2013). Osprey : protein design with ensembles, flexibility, and provable algorithms. *Methods in enzymology*, **523**, 87.

[48] Gasteiger, J. and Engel, T. (2006). *Chemoinformatics : a textbook*. John Wiley & Sons.

[49] Gasteiger, J. and Marsili, M. (1980). Iterative partial equalization of orbital electronegativity a rapid access to atomic charges. *Tetrahedron*, **36**(22), 3219 – 3228.

[50] Gasteiger, J. and Schulz, K.-P. (1990). The use of an associative memory system for the prediction of chemical reactivity. *Proceedings of International Conference on Fuzzy Logic & Neural Networks IIZUKA 90 Vol. 1*, **1**, 47–49.

[51] Gasteiger, J., Rudolph, C., and Sadowski, J. (1990). Automatic generation of 3d-atomic coordinates for organic molecules. *Tetrahedron Computer Methodology*, **3**(6), 537–547.

[52] Gasteiger, J., Li, X., Simon, V., Novič, M., and Zupan, J. (1993). Neural nets for mass and vibrational spectra. *Journal of molecular structure*, **292**, 141–159.

[53] Geist, A., Gropp, W. D., Huss-Lederman, S., Lumsdaine, A., Lusk, E. L., Saphir, W., Skjellum, A., and Snir, M. (1996). MPI-2 : Extending the message-passing interface. In L. Bougé, P. Fraigniaud, A. Mignotte, and Y. Robert, editors, *Proceedings of the 1st European Conference on Parallel and Distributed Computing (EuroPar'96)*, volume 1123 of *Lecture Notes in Computer Science*, pages 128–135. Springer.

[54] Ghemawat, S., Gobioff, H., and Leung, S.-T. (2003). The google file system. In *ACM SIGOPS operating systems review*, volume 37, pages 29–43. ACM.

[55] Gordon, D. B., Marshall, S. A., and Mayot, S. L. (1999). Energy functions for protein design. *Current Opinion in Structural Biology*, **9**(4), 509 – 513.

[56] Grant, J. A., Gallardo, M. A., and Pickup, B. T. (1996). A fast method of molecular shape comparison : A simple application of a Gaussian description of molecular shape. *Journal of Computational Chemistry*, **17**(14), 1653–1666.

[57] Gront, D. *et al.* (2011). Generalized fragment picking in Rosetta : design, protocols and applications. *PLoS ONE*, **6**(8), e23294.

[58] Grosso, A., Locatelli, M., and Pullan, W. (2008). Simple ingredients leading to very efficient heuristics for the maximum clique problem. *Journal of Heuristics*, **14**(6), 587–612.

[59] Guyon, F. and Tufféry, P. (2014). Fast protein fragment similarity scoring using a binet-cauchy kernel. *Bioinformatics*, **30**(6), 784–791.

[60] Guyon, F., Martz, F., Vavrusa, M., Bécot, J., Rey, J., and Tufféry, P. (2015). Bcsearch : fast structural fragment mining over large collections of protein structures. *Nucleic Acids Research*, **43**(W1), W378–W382.

[61] Halgren, T. A. (1996). Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *Journal of Computational Chemistry*, **17**(5), 490–519.

[62] Harder, T., Borg, M., Boomsma, W., Rogen, P., and Hamelryck, T. (2012). Fast large-scale clustering of protein structures using gauss integrals. *Bioinformatics*, **28**(4), 510–515.

[63] Hasegawa, A., Fujihara, Y., Morimoto, G., Hirano, Y., Okimoto, N., Taiji, M., and Funatsu, K. (2015). Development and advancement of a very-large-scale virtual compound library for drug discovery. *Proceedings of the Symposium on Chemoinformatics*, **2015**, 50–51.

[64] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Lear-*

ning : Data Mining, Inference, and Prediction, Second Edition. Springer Series in Statistics. Springer.

[65] Hawkins, P. C. D. and Nicholls, A. (2012). Conformer Generation with OMEGA : Learning from the Data Set and the Analysis of Failures. *Journal of Chemical Information and Modeling*, **52**(11), 2919–2936.

[66] Hert, J., Willett, P., Wilton, D. J., Acklin, P., Azzaoui, K., Jacoby, E., and Schuffenhauer, A. (2006). New methods for ligand-based virtual screening : use of data fusion and machine learning to enhance the effectiveness of similarity searching. *Journal of chemical information and modeling*, **46**(2), 462–470.

[67] Honig, B. and Nicholls, A. (1995). Classical electrostatics in biology and chemistry. *Science*, **268**(5214), 1144–1149.

[68] Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, **4**(4), 629–642.

[69] Horvath, D., Marcou, G., and Varnek, A. (2013). Do not hesitate to use tversky and other hints for successful active analogue searches with feature count descriptors. *Journal of chemical information and modeling*, **53**(7), 1543–1562.

[70] Hu, G., Kuang, G., Xiao, W., Li, W., Liu, G., and Tang, Y. (2012). Performance Evaluation of 2D Fingerprint and 3D Shape Similarity Methods in Virtual Screening. *Journal of Chemical Information and Modeling*, **52**(5), 1103–1113.

[71] Huang, N., Shoichet, B. K., and Irwin, J. J. (2006). Benchmarking Sets for Molecular Docking. *J. Med. Chem.*, **49**(23), 6789–801.

[72] Hudak, P., Peyton Jones, S., Wadler, P., Boutel, B., Fairbairn, J., Fasel, J., Guzmán, M. M., Hammond, K., Hughes, J., Johnsson, T., *et al.* (1992). Report on the programming language haskell : a non-strict, purely functional language version 1.2. *ACM SigPlan notices*, **27**(5), 1–164.

[73] Jaccard, P. (1901). Etude comparative de la distribution dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelle*, **4**.

[74] Jamroz, M. and Kolinski, A. (2013). Clusco : clustering and comparison of protein models. *BMC bioinformatics*, **14**(1), 62.

[75] Janert, P. K. (2010a). *Data analysis with open source tools*. " O'Reilly Media, Inc.".

[76] Janert, P. K. (2010b). *Gnuplot in action*. Manning.

[77] Jiang, L., Althoff, E. A., Clemente, F. R., Doyle, L., Röthlisberger, D., Zanghellini, A., Gallaher, J. L., Betker, J. L., Tanaka, F., Barbas, C. F., Hilvert, D., Houk, K. N., Stoddard, B. L., and Baker, D. (2008). De novo computational design of retro-aldol enzymes. *Science*, **319**(5868), 1387–1391.

[78] Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, **32**(3), 241–254.

[79] Kabsch, W. (1978). A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, **34**(5), 827–828.

[80] Kaufman, L. and Rousseeuw, P. (1987). *Clustering by means of medoids*. North-Holland.

[81] Kawabata, T. (2011). Build-Up Algorithm for Atomic Correspondence between Chemical Structures. *Journal of Chemical Information and Modeling*, **51**(8), 1775–1787.

[82] Kearsley, S. K. (1989). On the orthogonal transformation used for structural comparisons. *Acta Crystallographica Section A*, **45**(2), 208–210.

[83] Kendrew, J. C., Bodo, G., Dintzis, H. M., Parrish, R., Wyckoff, H., and Phillips, D. C. (1958). A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature*, **181**(4610), 662–666.

[84] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics*, **43**(1), 59–69.

[85] Kubinyi, H. (1998). Similarity and Dissimilarity : A Medicinal Chemist's View. *Perspectives in Drug Discovery and Design*, **9-11**(0), 225–252.

[86] Kuhlman, B., Dantas, G., Ireton, G. C., Varani, G., Stoddard, B. L., and Baker, D. (2003). Design of a novel globular protein fold with atomic-level accuracy. *Science*, **302**(5649), 1364–1368.

[87] Lagarde, N., Zagury, J.-F., and Montes, M. (2014a). Importance of the pharmacological profile of the bound ligand in enrichment on nuclear receptors : Toward the use of experimentally validated decoy ligands. *Journal of Chemical Information and Modeling*, **54**(10), 2915–2944. PMID : 25250508.

[88] Lagarde, N., Nasr, N. B., Jérémie, A., Guillemain, H., Laville, V., Labib, T., Zagury, J.-F., and Montes, M. (2014b). Nrlist bdb, the manually curated nuclear receptors ligands and structures benchmarking database. *Journal of Medicinal Chemistry*, **57**(7), 3117–3125. PMID : 24666037.

[89] Lagarde, N., Zagury, J.-F., and Montes, M. (2015). Benchmarking data sets for the evaluation of virtual ligand screening methods : Review and perspectives. *Journal of Chemical Information and Modeling*, **55**(7), 1297–1307.

[90] Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., *et al.* (2001). Initial sequencing and analysis of the human genome. *Nature*, **409**(6822), 860–921.

[91] Leaver-Fay, A. *et al.* (2011). Rosetta3 : An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. In *Methods in Enzymology*, volume 487, pages 545 – 574. Academic Press.

[92] Lee, J., Lee, D., Park, H., Coutsias, E. A., and Seok, C. (2010). Protein loop modeling by using fragment assembly and analytical loop closure. *Proteins : Structure, Function, and Bioinformatics*, **78**(16), 3428–3436.

[93] Lefranc, M.-P., Giudicelli, V., Ginestoux, C., Jabado-Michaloud, J., Folch, G., Bellahcene, F., Wu, Y., Gemrot, E., Brochet, X., Lane, J., *et al.* (2009). Imgt®, the international immunogenetics information system®. *Nucleic acids research*, **37**(suppl 1), D1006–D1012.

[94] Leroy, X. *et al.* (2012). *The OCaml system release 4.00 Documentation and user's manual*. INRIA, France.

[95] Levinthal, C. (1969). How to fold graciously. *Mossbauer spectroscopy in biological systems*, pages 22–24.

[96] Levitt, M. and Warshel, A. (1975). Computer simulation of protein folding. *Nature*, **253**(5494), 694–698.

[97] Li, H. and Zhou, Y. (2005). Scud : Fast structure clustering of decoys using reference state to remove overall rotation. *Journal of Computational Chemistry*, **26**(11), 1189–1192.

[98] Li, S. and Ng, Y. (2010). Calibur : a tool for clustering large numbers of protein decoys. *BMC Bioinformatics*, **11**(1), 25.

[99] Lipman, E. A., Schuler, B., Bakajin, O., and Eaton, W. A. (2003). Single-molecule measurement of protein folding kinetics. *Science*, **301**(5637), 1233–1235.

[100] Litzkow, M., Livny, M., and Mutka, M. (1988). Condor - a hunter of idle workstations. In *ICDCS'88*, pages 104–111. IEEE-CS Press.

[101] Looger, L. L., Dwyer, M. A., Smith, J. J., and Hellinga, H. W. (2003). Computational design of receptor and sensor proteins with novel functions. *Nature*, **423**(6936), 185–190.

[102] Luscombe, N. M., Greenbaum, D., Gerstein, M., *et al.* (2001). What is bioinformatics ? a proposed definition and overview of the field. *Methods of information in medicine*, **40**(4), 346–358.

[103] Mabrouk, M., Werner, T., Schneider, M., Putz, I., and Brock, O. (2015). Analysis of free modeling predictions by rbo aleph in casp11. *Proteins : Structure, Function, and Bioinformatics*, pages n/a–n/a.

[104] MacQueen, J. *et al.* (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[105] Mauri, A., Consonni, V., Pavan, M., and Todeschini, R. (2006). Dragon software : An easy approach to molecular descriptor calculations. *MATCH Commun. Math. Comput. Chem.*, **56**, 237–248.

[106] Millman, K. J. and Aivazis, M. (2011). Python for scientists and engineers. *Computing in Science and Engineering*, **13**(2), 9–12.

[107] Moreau, G. and Broto, P. (1980). The autocorrelation of a topological structure : a new molecular descriptor. *Nouveau journal de chimie*, **4**(6), 359 – 360.

[108] Moro, S., Bacilieri, M., Cacciari, B., and Spalluto, G. (2005). Autocorrelation of Molecular Electrostatic Potential Surface Properties Combined with Partial Least Squares Analysis as New Strategy for the Prediction of the Activity of Human A3 Adenosine Receptor Antagonists. *Journal of Medicinal Chemistry*, **48**(18), 5698–5704. PMID : 16134938.

[109] Mortier, W. J., Ghosh, S. K., and Shankar, S. (1986). Electronegativity-equalization method for the calculation of atomic charges in molecules. *Journal of the American Chemical Society*, **108**(15), 4315–4320.

[110] Moult, J., Fidelis, K., Kryshtafovych, A., Schwede, T., and Tramontano, A. (2014). Critical Assessment of methods of protein Structure Prediction (CASP)-round X. *Proteins : Structure, Function, and Bioinformatics*, **82**(S2), 1–6.

[111] Muchmore, S. W., Souers, A. J., and Akritopoulou-Zanze, I. (2006). The use of three-dimensional shape and electrostatic similarity searching in the identification of a melanin-concentrating hormone receptor 1 antagonist. *Chemical biology & drug design*, **67**(2), 174–176.

[112] Mulliken, R. S. (1955). Electronic population analysis on lcao–mo molecular wave functions. i. *The Journal of Chemical Physics*, **23**(10), 1833–1840.

[113] Mysinger, M. M., Carchia, M., Irwin, J. J., and Shoichet, B. K. (2012). Directory of Useful Decoys, Enhanced (DUD-E) : Better Ligands and Decoys for Better Benchmarking. *J. Med. Chem.*, **55**(14), 6582–94.

[114] Neudert, G. and Klebe, G. (2011). fconv : Format conversion, manipulation and feature computation of molecular data. *Bioinformatics*, **27**(7), 1021–2.

[115] Nicholls, A. and Honig, B. (1991). A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. *Journal of Computational Chemistry*, **12**(4), 435–445.

[116] O'Boyle, Noel and Banck, Michael and James, Craig and Morley, Chris and Vandermeersch, Tim and Hutchison, Geoffrey (2011). Open Babel : An open chemical toolbox. *Journal of Cheminformatics*, **3**(1), 33.

[117] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes 3rd Edition : The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA.

[118] R Core Team (2013). *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[119] Raman, S., Vernon, R., Thompson, J., Tyka, M., Sadreyev, R., Pei, J., Kim, D., Kellogg, E., , DiMaio, F., Lange, O., Kinch, L., Sheffler, W., Kim, B.-H., Das, R., Grishin, N. V., and Baker, D. (2009). Structure prediction for casp8 with all-atom refinement using rosetta. *Proteins : Structure, Function, and Bioinformatics*, **77**, 89–99.

[120] Rappe, A. K. and Goddard, W. A. (1991). Charge equilibration for molecular dynamics simulations. *The Journal of Physical Chemistry*, **95**(8), 3358–3363.

[121] Raymond, E. S. (2003). *The art of Unix programming*. Addison-Wesley Professional.

[122] Raymond, J. and Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, **16**(7), 521–533.

[123] Riniker, S. and Landrum, G. (2013). Open-source platform to benchmark fingerprints for ligand-based virtual screening. *Journal of Cheminformatics*, **5**(1), 26.

[124] Rodeh, O. and Teperman, A. (2003). zFS-a scalable distributed file system using object disks. In *Mass Storage Systems and Technologies, 2003.(MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on*, pages 207–218. IEEE.

[125] Rodriguez, D. D. *et al.* (2009). Crystallographic ab initio protein structure solution below atomic resolution. *Nat. Meth.*, **6**, 651–653.

[126] Roy, A. and Skolnick, J. (2015). Ligsift : an open-source tool for ligand structural alignment and virtual screening. *Bioinformatics*, **31**(4), 539–544.

[127] Saff, E. B. and Kuijlaars, A. B. (1997). Distributing many points on a sphere. *The mathematical intelligencer*, **19**(1), 5–11.

[128] Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., and Lyon, B. (1985). Design and implementation of the Sun network filesystem. In *Proceedings of the Summer USENIX conference*, pages 119–130.

[129] Satyanarayanan, M., Kistler, J. J., Kumar, P., Okasaki, M. E., Siegel, E. H., and Steere, D. C. (1990). Coda : A highly available file system for a distributed workstation environment. *Computers, IEEE Transactions on*, **39**(4), 447–459.

[130] Sayood, K. (2012). *Introduction to data compression*. Newnes.

[131] Schmuck, F. B. and Haskin, R. L. (2002). GPFS : A Shared-Disk File System for Large Computing Clusters. In *FAST*, volume 2, page 19.

[132] Schneider, G., Neidhart, W., Giller, T., and Schmid, G. (1999). Scaffold-hopping by topological pharmacophore search : A contribution to virtual screening. *Angewandte Chemie International Edition*, **38**(19), 2894–2896.

[133] Schuler, B. and Eaton, W. A. (2008). Protein folding studied by single-molecule {FRET}. *Current Opinion in Structural Biology*, **18**(1), 16 – 26. Folding and Binding / Protein-nucleic acid interactions.

[134] Sethaphong, L., Davis, J. K., Slabaugh, E., Singh, A., Haigler, C. H., and Yingling, Y. G. (2016). Prediction of the structures of the plant-specific regions of vascular plant cellulose synthases and correlated functional analysis. *Cellulose*, **23**(1), 145–161.

[135] Shapiro, M. (1977). The choice of reference points in best-match file searching. *Commun. ACM*, **20**(5), 339–343.

[136] Shehu, A., Clementi, C., and Kavraki, L. E. (2006). Modeling protein conformational ensembles : From missing loops to equilibrium fluctuations. *Proteins : Structure, Function, and Bioinformatics*, **65**(1), 164–179.

[137] Shen, Y., Maupetit, J., Derreumaux, P., and Tufféry, P. (2014). Improved pep-fold approach for peptide and miniprotein structure prediction. *Journal of Chemical Theory and Computation*, **10**(10), 4745–4758. PMID : 26588162.

[138] Shrestha, R. and Zhang, K. Y. J. (2014). Improving fragment quality for de novo structure prediction. *Proteins : Structure, Function, and Bioinformatics*, **82**(9), 2240–2252.

[139] Shrestha, R. and Zhang, K. Y. J. (2015). A fragmentation and reassembly method for *ab initio* phasing. *Acta Crystallographica Section D*, **71**(2), 304–312.

[140] Shrestha, R., Simoncini, D., and Zhang, K. Y. J. (2012). Error-estimation-guided rebuilding of *de novo* models increases the success rate of *ab initio* phasing. *Acta Crystallogr. Sect. D*, **68**(11), 1522–1534.

[141] Simoncini, D. *et al.* (2012). A Probabilistic Fragment-Based Protein Structure Prediction Algorithm. *PLoS ONE*, **7**(7), e38799.

[142] Simoncini, D., Allouche, D., de Givry, S., Delmas, C., Barbe, S., and Schiex, T. (2015a). Guaranteed discrete energy optimization on large protein design problems. *Journal of Chemical Theory and Computation*, **11**(12), 5980–5989.

[143] Simoncini, D., Nakata, H., Ogata, K., Nakamura, S., and Zhang, K. Y. (2015b). Quality assessment of predicted protein models using energies calculated by the fragment molecular orbital method. *Molecular Informatics*, **34**(2-3), 97–104.

[144] Slabaugh, E., Sethaphong, L., Xiao, C., Amick, J., Anderson, C. T., Haigler, C. H., and Yingling, Y. G. (2014). Computational and genetic evidence that different structural

conformations of a non-catalytic region affect the function of plant cellulose synthase. *Journal of Experimental Botany*, **65**(22), 6645–6653.

[145] Sneath, P. H. (1957). The application of computers to taxonomy. *Journal of general microbiology*, **17**(1), 201–226.

[146] Steipe, B. (2002). A revised proof of the metric properties of optimally superimposed vector sets. *Acta Crystallogr. Sect. A*, **58**(5), 506.

[147] Stephenson, N. (1999). *In the beginning... was the command line*. Avon books New York, NY.

[148] Sun, H.-P., Huang, Y., Wang, X.-F., Zhang, Y., and Shen, H.-B. (2015). Improving accuracy of protein contact prediction using balanced network deconvolution. *Proteins : Structure, Function, and Bioinformatics*, **83**(3), 485–496.

[149] Taminau, J., Thijs, G., and Winter, H. D. (2008). Pharao : Pharmacophore alignment and optimization. *Journal of Molecular Graphics and Modelling*, **27**(2), 161 – 169.

[150] Tange, O. (2011). GNU parallel-the command-line power tool. *The USENIX Magazine*, **36**(1), 42–47.

[151] Tanimoto, T. T. (1958). Elementary mathematical theory of classification and prediction. *IBM technical reports*.

[152] Teixeira, A. L. and Falcao, A. O. (2013). Noncontiguous Atom Matching Structural Similarity Function. *Journal of Chemical Information and Modeling*, **53**(10), 2511–2524.

[153] Tetko, I., Gasteiger, J., Todeschini, R., Mauri, A., Livingstone, D., Ertl, P., Palyulin, V., Radchenko, E., Zefirov, N., Makarenko, A., Tanchuk, V., and Prokopenko, V. (2005). Virtual Computational Chemistry Laboratory - Design and Description. *Journal of Computer-Aided Molecular Design*, **19**(6), 453–463.

[154] Theobald, D. L. (2005). Rapid calculation of rmsds using a quaternion-based characteristic polynomial. *Acta Crystallographica Section A*, **61**(4), 478–480.

[155] Thomas, D. and Hunt, A. (1999). *The Pragmatic Programmer : From Journeyman to Master*. Addison-Wesley Professional.

[156] Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, **63**(2), 411–423.

[157] Todeschini, R. and Consonni, V. (2009). *Molecular descriptors for chemoinformatics (2 volumes)*, volume 41. Wiley-VCH, Weinheim, Germany.

[158] Tramontano, A. and M, L. A. (1992). Common features of the conformations of antigen-binding loops in immunoglobulins and application to modeling loop conformations. *PROTEINS*, **13**, 231–245.

[159] Traoré, S., Allouche, D., André, I., de Givry, S., Katsirelos, G., Schiex, T., and Barbe, S. (2013). A new framework for computational protein design through cost function network optimization. *Bioinformatics*, **29**(17), 2129–2136.

[160] Tropsha, A. (2010). Best practices for qsar model development, validation, and exploitation. *Molecular Informatics*, **29**(6-7), 476–488.

[161] Tsai, H.-H. G., Tsai, C.-J., Ma, B., and Nussinov, R. (2004). In silico protein design by combinatorial assembly of protein building blocks. *Protein Science*, **13**(10), 2753–2765.

[162] Tsai, J., Taylor, R., Chothia, C., and Gerstein, M. (1999). The packing density in proteins : standard radii and volumes. *Journal of molecular biology*, **290**(1), 253–266.

[163] Tusar, M., Zupan, J., and Gasteiger, J. (1992). Neural networks and modelling in chemistry. *Journal de chimie physique*, **89**(7-8), 1517–1529.

[164] Tversky, A. (1977). Features of similarity. *Psychological review*, **84**(4), 327.

[165] Vanhee, P. *et al.* (2011). BriX : a database of protein building blocks for structural analysis, modeling and design. *Nucleic Acids Research*, **39**(suppl 1), D435–D442.

[166] Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., Smith, H. O., Yandell, M., Evans, C. A., Holt, R. A., *et al.* (2001). The sequence of the human genome. *science*, **291**(5507), 1304–1351.

[167] Voet, A. R., Kumar, A., Berenger, F., and Zhang, K. Y. (2014). Combining in silico and in cerebro approaches for virtual screening and pose prediction in sampl4. *Journal of computer-aided molecular design*, **28**(4), 363–373.

[168] Voordeckers, K., Brown, C. A., Vanneste, K., van der Zande, E., Voet, A., Maere, S., and Verstrepen, K. J. (2012). Reconstruction of ancestral metabolic enzymes reveals molecular mechanisms underlying evolutionary innovation through gene duplication. *PLoS Biol*, **10**(12), e1001446.

[169] Wagener, M., Sadowski, J., and Gasteiger, J. (1995). Autocorrelation of Molecular Surface Properties for Modeling Corticosteroid Binding Globulin and Cytosolic Ah Receptor Activity by Neural Networks. *Journal of the American Chemical Society*, **117**(29), 7769–7775.

[170] Wand, M. P. (1994). Fast Computation of Multivariate Kernel Estimators. *Journal of Computational and Graphical Statistics*, **3**(4), 433–445.

[171] Wang, Y., Backman, T. W. H., Horan, K., and Girke, T. (2013). fmcsr : mismatch tolerant maximum common substructure searching in r. *Bioinformatics*, **29**(21), 2792–2794.

[172] Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, **58**(301), 236–244.

[173] Wegner, J. K., Sterling, A., Guha, R., Bender, A., Faulon, J.-L., Hastings, J., O'Boyle, N., Overington, J., Van Vlijmen, H., and Willighagen, E. (2012). Cheminformatics. *Commun. ACM*, **55**(11), 65–75.

[174] Weininger, D. (1988). Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, **28**(1), 31–36.

[175] Wildman, S. A. and Crippen, G. M. (1999). Prediction of physicochemical parameters by atomic contributions. *Journal of chemical information and computer sciences*, **39**(5), 868–873.

[176] Willett, P. (2006). Similarity-based virtual screening using 2D fingerprints. *Drug Discovery Today*, **11**(23), 1046–1053.

[177] Williamson, M. P., Havel, T. F., and Wüthrich, K. (1985). Solution conformation of proteinase inhibitor iia from bull seminal plasma by 1 h nuclear magnetic resonance and distance geometry. *Journal of molecular biology*, **182**(2), 295–315.

[178] Wu, S., Skolnick, J., and Zhang, Y. (2007). Ab initio modeling of small proteins by iterative tasser simulations. *BMC Biology*, **5**(1), 17.

[179] Xu, D. *et al.* (2011). Automated protein structure modeling in CASP9 by I-TASSER pipeline combined with QUARK-based ab initio folding and FG-MD-based structure refinement. *PROTEINS*, pages 147 – 160.

[180] Xu, D. *et al.* (2012). Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. *PROTEINS*, **80**(7), 1715–1735.

[181] Yap, E.-H. and Head-Gordon, T. (2010). New and Efficient Poisson-Boltzmann Solver for Interaction of Multiple Proteins. *Journal of chemical theory and computation*, **6**(7), 2214–2224.

[182] Zhang, J. *et al.* (2011). Atomic-Level Protein Structure Refinement Using Fragment-Guided Molecular Dynamics Conformation Sampling. *Structure*, **19**(12), 1784 – 1795.

[183] Zhang, J. and Xu, D. (2013). Fast algorithm for population-based protein structural model analysis. *Proteomics*, **13**(2), 221–229.

[184] Zhang, Y. and Skolnick, J. (2004a). Scoring function for automated assessment of protein structure template quality. *Proteins : Structure, Function, and Bioinformatics*, **57**(4), 702–710.

[185] Zhang, Y. and Skolnick, J. (2004b). Spicker : A clustering approach to identify near-native protein folds. *Journal of Computational Chemistry*, **25**(6), 865–871.

[186] Zhang, Y., Arakaki, A. K., and Skolnick, J. (2005). Tasser : An automated method for the prediction of protein tertiary structures in casp6. *Proteins : Structure, Function, and Bioinformatics*, **61**(S7), 91–98.

[187] Zhou, J. and Wishart, D. S. (2013). An improved method to detect correct protein folds using partial clustering. *BMC bioinformatics*, **14**(1), 11.

[188] Zupan, J. and Gasteiger, J. (1993). *Neural networks for chemists : an introduction*. John Wiley & Sons, Inc.

Annexe A

Glossaire

AA : Acide Aminé. Une protéine est composée de plusieurs acides aminés. On dénombre 20 acides aminés standards.

ACPC : Auto Correlation of Partial Charges [15]. L'acronyme décrit ce que fait le cœur du logiciel. Il calcule la fonction d'autocorrélation sur les charges partielles de la molécule.

APBS : Adaptive Poisson-Boltzmann Solver [8]. Logiciel libre implémentant un solveur numérique de l'équation de Poisson-Boltzmann pour le calcul du champ électrostatique (entre autres).

ASC : Aire Sous la Courbe ROC. Équivalent français de l'acronyme anglais AUC défini ci-dessous.

AUC : Area Under the (ROC) Curve. L'aire sous la courbe ROC est une mesure de la performance d'un classificateur. Un classificateur parfait aura une AUC égale à 1.0. Un classificateur aléatoire (qui peut servir comme méthode témoin rapide à développer) aura une AUC proche de 0.5.

CASP : Critical Assessment of Structure Prediction. Une compétition organisée tous les deux ans pour évaluer l'état de l'art de la prédiction de structure de protéine en aveugle, c'est à dire en utilisant des protéines dont la structure n'est pas encore connue du public.

CATS : Chemically Advanced Template Search. Une méthode d'encodage de molécule rotation-translation invariante [40,132].

DEE : Dead End Elimination [34]. Un algorithme d'optimisation utilisé en protein

design pour trouver l'orientation des chaînes latérales parmi un ensemble de choix possibles discrets connus à l'avance.

DUD : Directory of Useful Decoys[71]. Un jeu de données pour les chercheurs en chémoinformatique et biologie structurale qui développent des méthodes de LBVS ou de docking.

DUDE : Directory of Useful Decoys Enhanced[113]. Version améliorée de DUD. Encore plus de protéines cibles, plus de ligands, plus de leurres et un protocole amélioré pour générer des leurres, plus difficiles à identifier, à partir des ligands.

FLOPS : Floating Point Operation per Second. Unité de mesure de la vitesse d'un ordinateur pour les opérations sur des nombres réels.

LBVS : Ligand-Based Virtual Screening. Exercice consistant à trouver des molécules actives sur une protéine cible (dont on n'aura pas forcément la structure) en fonction d'au moins une molécule active dont on connaît la formule chimique. La molécule active sert de « requête » pour trouver d'autres actives dans une base de données de molécules dont on ne connaît pas à priori l'activité. Après une campagne de LBVS, certaines molécules prometteuses verront leur activité testée en laboratoire. Si l'on connaît plusieurs molécules actives, on dispose de plus d'informations et la tâche est facilitée. On peut par exemple faire une requête consensus utilisant toutes les actives connues.

MACCS : Molecular ACCess System. Une empreinte de molécule sous forme d'un vecteur de bits. Chaque bit représente la réponse oui/non à une question posée sur la molécule. Par exemple, est-ce que la molécule a moins de trois atomes d'oxygène ? En informatique, on parle de fonction de hachage. Transformer une molécule en son empreinte MACCS est le résultat de l'application d'une fonction de hachage sur la molécule. C'est un encodage avec perte (tout comme ACPC) : la connaissance de l'empreinte ne permettra pas nécessairement de reconstruire la molécule et deux molécules différentes peuvent avoir la même empreinte.

MCS : Maximum Common Substructure (Search). Méthode qui permet d'identifier la composante connexe maximale commune aux graphes de deux molécules. Cf. Kawabata[81] pour une méthode rapide utilisant des heuristiques.

MPI : Message Passing Interface. Librairie de programmation pour le calcul distribué haute performance. Ses primitives sont relativement bas niveau.

NFS : Network File System. Un système de fichier centralisé (créé par l'entreprise SUN [128] en 1985) qui passe par le réseau. Quel que soit l'ordinateur que l'on utilise, avec NFS on a l'impression que les données sont stockées localement alors qu'elles sont accédées via le réseau sur un serveur distant. NFS est pratique à l'utilisation mais son architecture n'est pas adaptée au calcul distribué data-intensif et il ne protège pas la confidentialité des fichiers de l'utilisateur.

PDB : Protein Data Bank. Base de données publique et mondiale des protéines dont la structure est connue [1]. PDB:1M6T signifie la protéine de code PDB 1M6T. Pour les novices, on peut recommander la lecture de l'article « Molecule of the Month » qui parait chaque mois pour mettre en avant une protéine particulièrement intéressante ; article qui s'accompagne toujours d'une illustration magnifique.

PEOE : (Iterative) Partial Equalization of Orbital Electronegativity. Méthode de calcul rapide des charges partielles des atomes d'une molécule, développée par Johann Gasteiger et Mario Marsili [49]. Elle est aussi souvent nommée modèle de Gasteiger-Marsili.

QCP : Quaternion-based Characteristic Polynomial. Une méthode efficace de calcul du RMSD [154].

QSAR (modelling) : Quantitative Structure Activity Relationship. Développement de modèles pour explorer et exploiter des bases de données de molécules actives [160]. Un bon modèle de QSAR peut prédire l'activité d'une molécule qu'il n'a jamais examinée auparavant. Le développement de modèles de QSAR est un domaine d'application de l'apprentissage artificiel.

RICC : RIKEN Integrated Cluster of Clusters. Le supercalculateur RICC (campus RIKEN de Wakoushi) était le plus gros supercalculateur du Japon en juin 2009.

RIKEN : Centre de recherche scientifique national japonais. À l'origine, l'institut était spécialisé en chimie et physique. Il est maintenait plus diversifié [2].

1. `http://www.rcsb.org/pdb/home/home.do`
2. `www.riken.jp/en`

RMSD : Root Mean Square Deviation. Distance très utilisée en biologie structurale, mesurée entre deux conformations de protéines après leur superposition optimale.

ROC : Receiver Operating Characteristic (Curve). Courbe résultant du tracé du taux de vrai positif (le classificateur a dit vrai et c'est la réponse correcte) en fonction du taux de faux positif (le classificateur a dit vrai mais c'est une mauvaise réponse) pour un classificateur binaire.

SCUD : Structure ClUstering of Decoys[97]. Logiciel pour faire du clustering de modèles de protéines. SCUD utilise le RMSD_{ref} au lieu du RMSD classique. C'est une borne haute du RMSD qui est moins coûteuse à calculer. En effet, toutes les molécules sont superposées à une seule au lieu d'être superposées deux à deux à la volée. Cf. l'article de Li et Ng[98] pour des bornes hautes et basses du RMSD peu coûteuses à calculer.

Annexe B

Logiciel libre

Au cours de nos travaux, nous avons aussi contribué de nombreux rapports de bugs et demandes de fonctionnalités concernant les logiciels libres que nous utilisions. Nous avons aussi interagi avec la communauté d'utilisateurs et de développeurs sur les listes e-mail de ces même logiciels. On peut être un informaticien isolé dans une équipe de recherche mais recevoir une aide non négligeable grâce à ces communautés. Nous avons aussi empaqueté de nombreux logiciels et librairies OCaml afin qu'ils soient installâbles automatiquement [1].

Outre les logiciels présentés précédemment (PAR, Durandal, EleKit, ACPC et Fragger), nous listons ici plusieurs logiciels et librairies écrits en OCaml qui ont été réalisés au cours de cette thèse et mis à disposition du public. Ce sont des composants qui ont été externalisés de nos logiciels afin de les rendre utiles au plus grand nombre.

1. Un patch à Open Babel afin de travailler avec des molécules dans l'espace hydrophobe [2]. Open Babel sait calculer la contribution à LogP des atomes lourds en utilisant le modèle de Wildman et Crippen [175], mais il n'était pas possible d'extraire cette information en dehors d'Open Babel. Open Babel ajoute les contributions des atomes d'hydrogènes aux atomes lourds auxquels ils sont attachés. Nous avons donc patché Open Babel afin de créer un fichier qui liste la position des atomes lourds d'une molécule ainsi que la contribution estimée à LogP pour chacun. Ce patch permet à ACPC de travailler avec des molécules dans l'espace hydrophobe.
2. dolog (`https://github.com/UnixJunkie/dolog`) : un logger paresseux pour OCaml,

1. `https://opam.ocaml.org/packages/`
2. `https://github.com/UnixJunkie/openbabel/commits/logP_contrib_per_heavy_atom`

utilisé dans presque tous les projets mentionnés ici. Un logger paresseux est un logger qui ne formate pas les messages qui n'ont pas un niveau de criticité suffisant pour être imprimé. Les logiciels scientifiques impriment souvent des nombres dans leurs messages, mais mettre un nombre en forme avant de l'afficher a un coût non négligeable. Dolog à reçu des améliorations de code et des nouvelles fonctionnalités venant de quatre autres programmeurs, un succès pour un logiciel libre.

3. interval-tree (`https://github.com/UnixJunkie/interval-tree`) : une librairie pour créer puis interroger de manière très efficace un arbre d'intervalles de nombres réels[33]. La librairie interval-tree est utilisée pour accélérer les calculs dans EleKit (calculs de collisions exactes de sphères minimisés grâce à des boites englobantes cubiques) et dans ACPC (détermination rapide des triangles qui se chevauchent lors du scoring de deux molécules).

4. vector3 (`https://github.com/UnixJunkie/vector3`) : une librairie pour manipuler des vecteurs 3D en OCaml. Cette librairie est nécessaire au logiciel EleKit qui manipule un grand nombre de points en 3D lors d'une comparaison de molécules. En effet, les molécules sont discrétisées dans une grille 3D fine par le logiciel APBS qui calcule le champ électrostatique.

5. genspir (`https://github.com/UnixJunkie/genspir`) : une implémentation de la spirale généralisée[127]. Une méthode qui permet d'échantillonner de manière presque régulière la surface d'une sphère avec des points. Cette librairie est utilisée dans EleKit pour surfacer les atomes d'une molécule.

le cnam

François Bérenger

Nouveaux logiciels pour la biologie structurale computationnelle et la chémoinformatique

le cnam

Résumé : cette thèse introduit cinq logiciels de trois domaines distincts : le calcul parallèle et distribué, la bioinformatique structurale et la chémoinformatique. Le logiciel pour le calcul parallèle et distribué s'appelle PAR. PAR permet d'exécuter des expériences computationnelles indépendantes de manière parallèle et distribuée. Les logiciels pour la bioinformatique structurale sont Durandal, EleKit et Fragger. Durandal exploite la propagation de contraintes géométriques afin d'accélérer l'algorithme de partitionnement exact pour des conformations de protéines. EleKit permet de mesurer la similarité électrostatique entre une petite molécule et la protéine qu'elle est conçue pour remplacer sur une interface protéine-protéine. Fragger est un cueilleur de fragments de protéines permettant de sélectionner des fragments dans la banque de protéines mondiale. Enfin, le logiciel de chémoinformatique est ACPC. ACPC permet l'encodage fin, d'une manière rotation-translation invariante, d'une molécule dans un ou une combinaison des trois espaces chimiques (électrostatique, stérique ou hydrophobe). ACPC est un outil de criblage virtuel qui supporte les requêtes consensus, l'annotation de la molécule requête et les processeurs multi-cœurs.
Mots clés : regroupement, clustering, protéine, similarité électrostatique, autocorrélation, criblage virtuel, ligand, programmation parallèle, programmation distribuée.

Abstract : this thesis introduces five software useful in three different areas : parallel and distributed computing, structural bioinformatics and chemoinformatics. The software for parallel and distributed computing is PAR. PAR allows to execute independent computational experiments in a parallel and distributed way. The software for structural bioinformatics are Durandal, EleKit and Fragger. Durandal exploits the propagation of geometric constraints to accelerate the exact clustering algorithm for protein conformations. EleKit allows to measure the electrostatic similarity between a chemical molecule and the protein it is designed to replace at a protein-protein interface. Fragger is a fragment picker able to select protein fragments in the whole protein data-bank. Finally, the chemoinformatics software is ACPC. ACPC encodes in a rotation-translation invariant way a molecule in any or a combination of three chemical spaces (electrostatic, steric or hydrophobic). ACPC is a ligand-based virtual screening tool supporting consensus queries, query molecule annotation and multi-core computers.
Keywords : clustering, protein, electrostatic similarity, autocorrelation, virtual screening, ligand, parallel programming, distributed programming.