Université Pierre et Marie Curie
École Doctorale iViv
Interdisciplinaire pour le vivant
Spécialité : Biophysique moléculaire

Thèse en vue de l'obtention du diplôme de
docteur de l'université Pierre et Marie Curie

Directeur de thèse : Dr Gildas Bertho
codirection : Pr. Jean-Pierre Girault

Thèse de doctorat :

Étude d'inhibiteurs de l'intégrase du VIH-1 par RMN et modélisation moléculaire.

Développement et validation d'un outil de criblage virtuel.

Guillaume Bouvier
<guillaume.bouvier@parisdescartes.fr>

Rapporteurs : Pr. Jean-Marc Lancelin
Dr Michael Nilges
Examinateurs : Dr Richard Benarous
Pr. Germain Trugnan

Paris, le 22 juillet 2010

Remerciements

Je tiens à exprimer tout d'abord mes remerciements au Dr Gildas Bertho pour avoir accepté de diriger cette thèse. Les nombreuses discussions scientifiques que nous avons partagées m'ont été d'une aide précieuse dans le déroulement de ces trois années de thèse.
Merci au Pr. Jean-Pierre Girault, directeur puis co-directeur de ma thèse, qui m'a accueilli au sein de l'équipe de RMN et modélisation moléculaire du laboratoire de chimie et de biochimie pharmacologiques et toxicologiques (UMR8601 – CNRS-Université Paris Descartes). Je le remercie pour ses conseils précieux qu'il m'a donnés tout au long de ma thèse.
Un grand merci au Dr Nathalie Evrard-Todeschi, pour son dynamisme, son soutien et sa confiance. Je lui exprime ma reconnaissance pour ses qualités humaines et scientifiques.
Mes remerciements s'adressent également au Dr Richard Benarous, directeur scientifique de la société CellVir. Nos rencontres scientifiques, qui ont ponctué ma thèse, ont toujours étaient très enrichissantes et un véritable moteur pour l'avancée du sujet.
Merci au Dr Sabine Barbey, chimiste de la société CellVir, sans qui les molécules ne seraient restées que virtuelles.
Je tiens aussi à remercier Damien Bonnard, ingénieur de recherche de la société CellVir, pour m'avoir fourni les protéines étudiées.
Merci à Luc Tamisier pour ses conseils avisés concernant la mise en ligne du logiciel AuPosSOM, ce qui a permis de le rendre accessible à tous.
Mes sincères remerciements envers les membres du jury, qui ont accepté de juger mon travail de thèse et en particulier au Pr. Jean-Marc Lancelin et au Dr Michael Nilges, rapporteurs de mes travaux de thèse, ainsi qu'au Pr. Germain Trugnan, directeur de l'école doctorale iViv.

Résumé

Les anti-intégrases actuels sont dirigés contre l'activité catalytique de l'intégrase du VIH-1. Cependant, le développement de ces molécules inhibitrices de l'activité catalytique est rendu difficile du fait de l'apparition rapide de mutants résistants, d'une part, et de l'inhibition non souhaitée de certaines enzymes cellulaires de recombinaison d'autre part.

La découverte de l'importance de l'interaction entre la protéine LEDGF (*Lens Epithelial Derived Growth Factor*) et l'intégrase, dans le cycle viral, a permis de dévoiler une cible potentielle pour le développement de nouveaux anti-intégrase.

Une approche de criblage virtuel par *docking* a été utilisée afin de découvrir de nouvelles molécules pouvant potentiellement inhiber l'interaction intégrase·-LEDGF. Les études *in silico* sont menées en parallèle avec des expériences RMN de type STD (*Saturation Transfer Difference*) et WaterLOGSY (*Water-Ligand Observed via Gradient SpectroscopY*), afin d'identifier les interactions entre les hits potentiels et le complexe intégrase·LEDGF.

L'étude RMN des interactions ligand·protéine a permis de valider la fixation de ces nouvelles molécules sur l'intégrase. Les expériences de compétitions ont permis de montrer que certains hits ne se fixent pas au niveau du site catalytique. Le développement et la validation d'un nouvel outil de criblage virtuel : le logiciel AuPosSOM, nous a permis de caractériser le mode de fixation de ces nouvelles molécules anti-intégrase.

Abstract

Current anti-integrase agents targets the catalytic activity of the HIV-1 integrase. However, their development is hampered by the rapid onset of resistance mutations as well as the unwanted inhibition of cellular recombining enzymes.
The LEDGF (Lens Epithelial Derived Growth Factor) has been identified as an important interacting factor of the integrase in the viral cycle. Therefore, the disruption of this interaction can be considered as a new potential target for antiretroviral agents.
A structure-based virtual screening approach has been used to discover new potential inhibitors of the integrase·LEDGF interaction. The combination of in silico *studies with NMR experiments, like STD (Saturation Transfer Difference) and WaterLOGSY (Water-Ligand Observed via Gradient SpectroscopY), has allowed the characterization of the interactions between ligands and integrase·LEDGF complex.*
The NMR studies validated the binding of those new ligands to the integrase. NMR competition experiments led to the identification of non-catalytic ligands. A new tool for virtual screening – the AuPosSOM software – has been developed and validated. This software allowed us to characterize the binding mode of these new anti-integrase agents.

Table des matières

Première partie

Introduction

Chapitre 1

Le Virus de l'immunodéficience humaine : cycle viral et stratégies antirétrovirales

1.1 Introduction

Le VIH [1] est l'agent étiologique du SIDA [2]. Le VIH-1 est un *Lentivirus* (membre de la famille des rétrovirus) à ARNs [3] simples brins. Il infecte spécifiquement les lymphocytes T $CD4^+$, les macrophages et les cellules dendritiques du système immunitaire humain et provoque leur mort. Lorsque le nombre de cellules décline en dessous d'une valeur critique, l'immunité à médiation cellulaire est fortement diminuée voire annihilée, laissant la place à des infections dîtes opportunistes. Le SIDA fût pour la première fois reporté par l'U.S.-CDC [4] en 1981. C'est en 1983 que l'équipe du Professeur Jean-Claude Chermann de l'Institut Pasteur, sous la direction de Luc Mantagnier, découvre et isole son agent étiologique : le VIH [7]. Cette menace pour la santé humaine s'est rapidement transformée en une épidémie mondiale faisant plus de 25 millions de mort de 1981 à 2006. Le SIDA représente de nos jours la troisième cause de décès au rang des maladies infectieuses.

Deux types de virus de l'immunodéficience humaine existe : le VIH-1 et le VIH-2. Le VIH-2 est moins transmissible et la période entre l'infection et la déclaration de la maladie associée est plus longue que pour le VIH-1. À l'échelle mondiale, le VIH-2 est moins répandue que le VIH-1 et sa

1. Virus de l'Immunodéficience Humaine
2. Syndrome d'Immunodéficience Acquise
3. Acides RiboNucléiques
4. *United States - Centers for Disease Control and prevention*

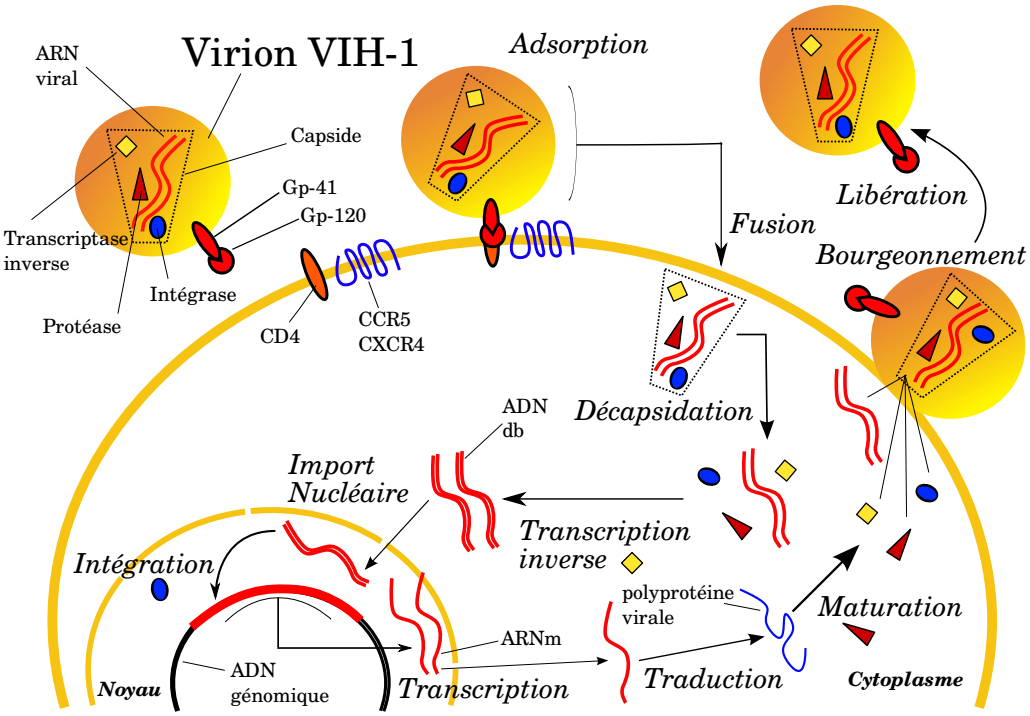


FIGURE 1.1 – Cycle de réplication du VIH-1

distribution se localise essentiellement en Afrique de l'ouest. Le VIH-1 est le type prédominant.

1.2 Cycle de réplication du VIH-1

Le VIH-1 utilise la machinerie cellulaire dans un grand nombre d'étape du processus de réplication. Cependant le cycle de réplication (voir figure 1.1) fait intervenir des réactions propres au virus que la cellule ne peut pas réalisées. Ces étapes doivent être catalysées par des enzymes d'origine virale. Trois enzymes possèdent des activités catalytiques essentielles à la réplication du virus : la transcriptase inverse, l'intégrase et la protéase. Ces enzymes se trouvent au sein du virus, protégées par la capside et la membrane virales. L'ensemble des étapes nécessaires à la réplication du virus est présenté sur la figure 1.1 et est détaillé ci dessous :

Adsorption et fusion L'infection cellulaire débute par l'interaction entre la protéine virale gp120 et des récepteurs extra-cellulaires. Le récepteur cellulaire principale du VIH-1 est le récepteur CD4 et les protéines

trans-membranaires CCR5 et CXCR4 jouent le rôle de co-récepteur [28]. L'étape d'adsorption est suivie par le phénomène de fusion de la membrane virale avec la membrane cellulaire. Le matériel viral, enzymes et ARNs, est ainsi libéré dans le cytoplasme de la cellule infectée après dissociation de la capside (décapsidation).

Transcription inverse La transcriptase inverse, une enzyme virale, catalyse la réaction de transcription inverse qui permet le passage de l'ARN viral à l'ADN [5].

Intégration L'ADN viral nouvellement synthétisé est intégré au sein du génome cellulaire. L'enzyme virale qui catalyse cette réaction est l'intégrase. Le processus d'intégration est complexe et nécessite en amont l'import nucléaire du matériel viral. Cette étape clé du cycle viral sera détaillé dans la suite, ainsi que les mécanismes mis en jeu. L'ADN viral intégré est appelé provirus.

Transcription La transcription est l'étape qui permet de produire des ARNs messagers (ARNm) à partir de l'ADN proviral. L'ensemble des mécanismes impliqués est réalisé par la machinerie cellulaire qui est ainsi détournée par le virus.

Traduction Cette étape de production des protéines virales à partir des informations de séquence portées par les ARNm met aussi en jeu la machinerie cellulaire. Les protéines produites sont des polyprotéines virales qui ont besoin d'être maturées afin de produire les protéines virales fonctionnelles.

Maturation La maturation consiste en la coupure spécifique de liaisons peptidiques des polyprotéines virales ce qui permet la production des protéines virales matures.

Bourgeonnement Le bourgeonnement des nouveaux virus permet la libération de virus actifs pouvant infecter de nouvelles cellules. Les glycoprotéines virales (gp120 et gp41), exprimées à la surface de la cellule infectée, sont incorporées à l'enveloppe virale provenant de la cellule infectée.

1.3 Les agents anti-VIH

Il n'existe pas encore de vaccin protégeant efficacement contre l'infection par le VIH-1 [89]. Cependant, une étude récente a montré une protection partielle par vaccination contre le VIH-1 chez l'homme [150].

5. Acide DésoxyriboNucléique

Des progrès importants ont été réalisés dans le traitement des malades du SIDA. La thérapie développée, nommée HAART [6], consiste en la combinaison de différentes molécules agissant sur des cibles moléculaires différentes. Les buts de cette thérapie est d'améliorer la qualité de vie des patients, de réduire les complications et de réduire la virémie en dessous des seuils de détection. Cependant ces traitements ne soignent pas le patient de manière définitive, l'arrêt du traitement entraînant la ré-apparition des symptômes de la maladie.

La mise en place d'un traitement curatif est difficile du fait de l'intégration du génome viral, sous forme de provirus, dans le génome de la cellule infectée. De plus le VIH-1 est un virus qui mute de manière importante, ce qui entraîne rapidement des phénomènes de résistance aux molécules développées. Ce fort taux de mutation est expliqué par le *turnover* réplicatif élevé du virus combiné à la faible fidélité de la transcriptase inverse lors du phénomène de synthèse de l'ADN viral à partir de l'ARN viral.

La combinaison de différentes molécules anti-VIH utilisée au cours de la thérapie HAART permet de contrebalancer les phénomènes de résistance du virus. Classiquement, les cibles privilégiées des thérapies mises en place sont la transcriptase inverse et la protéase du VIH-1 [119].

De manière générale, les molécules anti-virales peuvent agir sur les protéines virales ou cellulaires. La première approche permet de développer des molécules plus spécifiques, moins toxiques, mais entraînant des phénomènes de résistance de la part du virus. La deuxième approche permet d'avoir un spectre d'action plus large suivant les souches virales, évite les phénomènes de résistance, mais entraîne une toxicité importante [119].

Depuis 1987 [124], les stratégies développées pour lutter contre le VIH-1 sont définies comme indiquées dans le tableau 1.1 page ci-contre. L'ensemble des étapes du cycle peuvent servir de cible thérapeutique potentielle pour le développement des anti-VIH. Actuellement, plus de 30 molécules et combinaisons de molécules différentes sont autorisées par la FDA [7] [48]. Ces molécules appartiennent à six classes différentes détaillées ci dessous.

Inhibiteurs nucléosidiques de la transcriptase inverse (NRTIs [8])

La première molécule développée pour le traitement du SIDA est le NRTI 3'-azido-2',3'-dideoxythymidine (AZT) [125], qui est un analogue nucléosidique, inhibant le fonctionnement de la transcriptase inverse. Toutes les molécules de cette classe agissent en inhibant l'activité catalytique de la transcriptase inverse. Ce sont des analogues du substrat

6. *Highly Active Anti-Retroviral Therapy*
7. *United States - Food and Drug Administration*
8. *Nucleoside Reverse Transcriptase Inhibitors*

Étape du cycle virale	Mode d'action potentiel
Adsorption à la cellule hôte et entrée des virus	Liaison d'une molécule ou d'un anticorps à la protéine gp120. Blocage des co-récepteurs CXCR4 et CCR5
Fusion des membranes virales et cellulaires	Liaison d'une molécule à la protéine gp41
Transcription inverse de l'ARN en ADN	Inhibition de la transcriptase inverse
Dégradation de l'ARN viral dans l'hybride ARN·ADN	Inhibition de la RNase H
Intégration de l'ADN viral	Inhibition de l'intégrase
Expression des gènes viraux	Inhibiteurs de la protéine virale tat
Maturation des virions	Inhibiteurs de la protéase, de la myristoylation et de la glycosylation
Bourgeonnement des virus	Interférons

TABLE 1.1 – Mode d'action et cibles thérapeutiques potentielles pour le développement d'anti-VIH. D'apès Mitsuya *et al.* [124].

de cette enzyme.

Inhibiteurs non nucléosidiques de la transcriptase inverse (NNRTIs[9])
Les première molécules inhibitrices de la transcriptase inverse, non analogue du substrat, sont apparues dans les années 1989/1990 [5, 121, 140]. Ces inhibiteurs agissent en se liant de manière non compétitive – par rapport au substrat de l'enzyme – à un site allostérique spatialement proche ($\sim$ 15 Å) du site catalytique [34].

Inhibiteurs de la protéase (PIs[10])
Ces inhibiteurs bloquent l'étape tardive de maturation des polyprotéines virales. Ces inhibiteurs empêchent donc la formation de particules virales infectieuses.

Inhibiteurs de la fusion (FIs[11])
La glycoprotéine virale gp120 se lie au récepteur CD4 de la cellule hôte. Cette interaction est à l'origine d'une modification conformationnelle qui permet de dévoiler le site de fixation de gp120 aux co-récepteurs CXCR4 et CCR5. Cette seconde interaction modifie la conformation de la protéine gp41 qui devient alors active. Ceci déclenche alors la fusion de l'enveloppe virale avec la membrane cellulaire. Les inhibiteurs de fusion comme l'enfuvirtide [113], le seul composé autorisé dans le traitement du SIDA, vise la glycoprotéine virale gp41.

Inhibiteurs de l'entrée - Inhibiteurs des co-récepteurs (CRIs[12])
Du fait de l'importance des co-récepteurs CCR5 et CXCR4 dans le mécanisme d'entrée du VIH-1 dans la cellule hôte, des inhibiteurs de cette étape ont été développés. Le composé maraviroc [38] est la seule molécule antagoniste de CCR5 autorisée pour le traitement du SIDA.

Inhibiteurs du transfert de brin de l'intégrase (INSTIs[13])
Ces inhibiteurs agissent sur l'étape d'intégration de l'ADN viral au sein de l'ADN cellulaire de l'hôte. Ils vont être décrit plus en détail par la suite.

9. *Nonnucleoside Reverse Transcriptase Inhibitors*
10. *Protease Inhibitors*
11. *Fusion Inhibitors*
12. *Co-Receptors Inhibitors*
13. *Integrase Strand Transfer Inhibitors*

Chapitre 2

L'intégrase du VIH-1 : Aspects structuraux et fonctionnels

2.1 Les activités catalytiques de l'intégrase

L'intégrase catalyse l'insertion covalente d'une copie de l'ADN viral, produit de la transcription inverse de l'ARN viral génomique, dans le génome de la cellule infectée (voir figure 1.1 page 16). Les fonctions catalytiques de l'intégrase sont essentielles au cycle de réplication du VIH et ont donc fait l'objet d'une recherche pharmacologique intense.

L'intégrase possède deux activités catalytiques essentielles au cycle de réplication du virus : le *3'-processing* et le transfert de brins. Les mécanismes mis en jeu sont détaillés sur la figure 2.1 page suivante

L'ADN viral intégré au sein du génome cellulaire est un ADN linéaire issu de la transcription inverse du génome viral. Dans un premier temps, l'intégrase se fixe sur une séquence courte située aux deux extrémités des répétitions terminales de l'ADN viral (LTR [1]). Elle catalyse alors une réaction endonucléolytique des extrémités 3'-OH de l'ADN viral qui entraîne l'élimination d'un dinucléotide à chaque extrémité. Cette réaction est communément appelée *3'-processing* (voir figure 2.1 page suivante).

L'intégration de l'ADN viral dans le génome cellulaire correspond, au sens strict, à une réaction de transfert de brins. Cette réaction, aussi catalysée par l'intégrase, correspond à la formation concomitante de deux liaisons covalentes par attaque nucléophile des deux extrémités 3'-OH de l'ADN viral sur deux groupements phosphate, situé à 5 paires de bases d'intervalle, de l'ADN cellulaire. La structure formée présente alors des bases non appariées (en rouge sur la figure 2.1 page suivante) correspondant aux deux bases de

1. *Long Terminal Repeat*

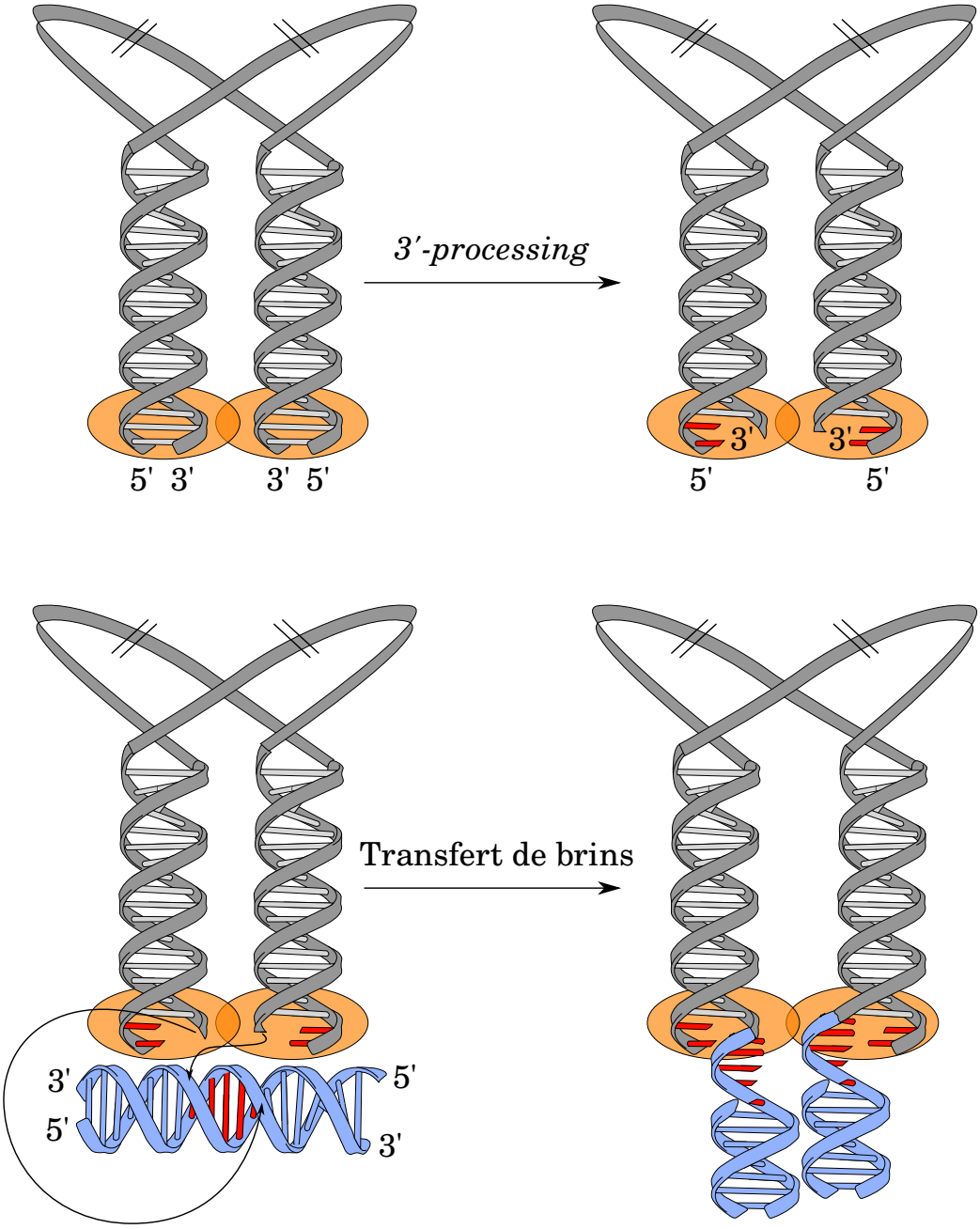


FIGURE 2.1 – Processus d'intégration du génome viral (en gris) au sein du génome cellulaire (en bleu) catalysé par l'intégrase (en orange). Les bases non appariées sont indiquées en rouge. D'après Pommier *et al.* [143].

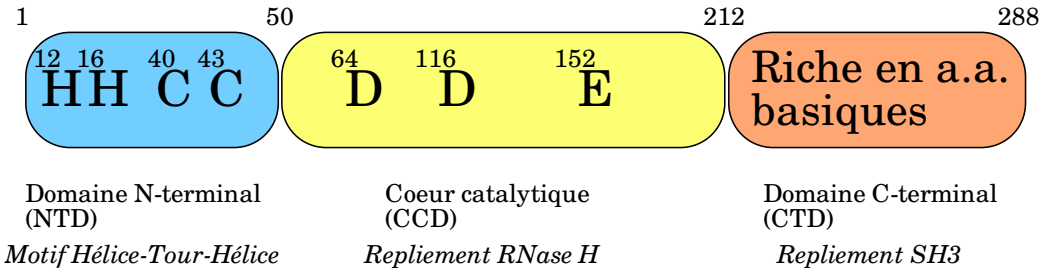


FIGURE 2.2 – Les trois domaines de l'intégrase. Les résidus HHCC et DDE indiqués sont impliqués dans la fixation de cations métalliques. L'organisation topologique de chaque domaine est indiqué par le texte en italique.

l'extrémité 5' de l'ADN viral et aux cinq bases non appariées générées lors de l'étape de transfert de brins. La réparation de ces structures non appariées est réalisée par des enzymes de réparation de l'ADN de la cellule infectée [143].

In vivo, la maturation des extrémité 3'-OH – *3'-processing* – et le transfert de brin sont spatialement et temporellement séparés. En effet le *3'-processing* a lieu dans le cytoplasme de la cellule, tandis que l'intégration prend place, évidemment, dans le noyau de la cellule infectée.

In vitro une troisième réaction peut être mis en évidence. Cette réaction nommée désintégration est la réaction inverse du transfert de brin [26]. Contrairement à la réaction de maturation des extrémités 3'-OH et au transfert de brins qui requièrent les trois domaines de l'intégrase, la réaction de désintégration peut être catalysée uniquement par un domaine isolé de la protéine qui contient le site actif : le CCD [2] [53].

2.2 Structure de l'intégrase

L'intégrase est une protéine de 288 acides aminés (32 kDa). Elle comporte trois domaines indépendants (voir figure 2.2) : un domaine central (50-212), responsable de l'activité catalytique, un domaine N-terminal (1-49) qui est nécessaire à l'oligomérisation de l'intégrase, enfin un domaine C-terminal (213-288) qui possède la propriété de se fixer à l'ADN de façon non spécifique.

Jusqu'à ce jour, aucune structure complète de l'intégrase du VIH-1 entière (1-288) n'a été résolue pour des raisons de faible solubilité et de flexibilité interdomaine. En revanche, plusieurs structures des domaines isolés ou bien

2. *Core Catalytic Domain*

encore de deux domaines consécutifs ont été élucidées. Cependant Cherepanov *et al.* ont publié récemment, dans le journal Nature, la structure tridimensionnelle de l'intégrase du PFV [3] en complexe avec son ADN viral. L'alignement séquentiel et structural de l'intégrase du PFV avec les structures des domaines obtenus pour l'intégrase du VIH-1 montre une grande conservation des éléments structuraux principaux. Ce résultat est majeur pour la compréhension du fonctionnement des anti-intégrases actuels, puisque la structure de cette intégrase a aussi était obtenue en présence des deux anti-intégrases les plus aboutit, à savoir le raltégravir (MK0518) et l'elvitegravir (GS9137). Cette structure peut donc servir de base pour le développement de nouveaux inhibiteurs de l'intégrase du VIH-1.

2.2.1 Domaine N-terminal

Le domaine N-terminal (NTD [4]) comporte un motif HHCC, analogue à un doigt de zinc. Il fixe effectivement ce cation et sa fixation induit le repliement du domaine N-terminal, la multimérisation de l'intégrase entière et amplifie l'activité catalytique de l'intégrase [181]. Chaque unité monomérique est structurée en quatre hélices α, les hélices $\alpha 2$ et $\alpha 3$ formant un motif hélice-tour-hélice. L'ensemble est stabilisé par la coordination du zinc et par des interactions hydrophobes.

2.2.2 Domaine C-terminal

Le domaine C-terminal (CTD [5]) est constitué de cinq brins β antiparallèles, structure en tonneau β, et possède une topologie comparable au domaine SH3 [6]. Le CTD établie des interactions faibles non spécifiques avec le squelette phosphodiester de l'ADN viral. Le CTD pourrait aussi être impliqué dans la fixation de l'ADN cellulaire cible [62].

2.2.3 Domaine catalytique

Le domaine central, ou CCD, contient le site actif constitué d'une triade catalytique regroupant les trois acides aminés D64, D116 et E152. Chacun des trois acides aminés est strictement requis pour l'activité catalytique de la protéine. De nombreuses structures cristallographiques du domaine catalytique de l'intégrase du VIH-1 ont été obtenues. Cependant toutes ces

3. *Prototype Foamy Virus*
4. *N-terminal domain*
5. *C-terminal domain*
6. *SRC Homology 3 Domain*

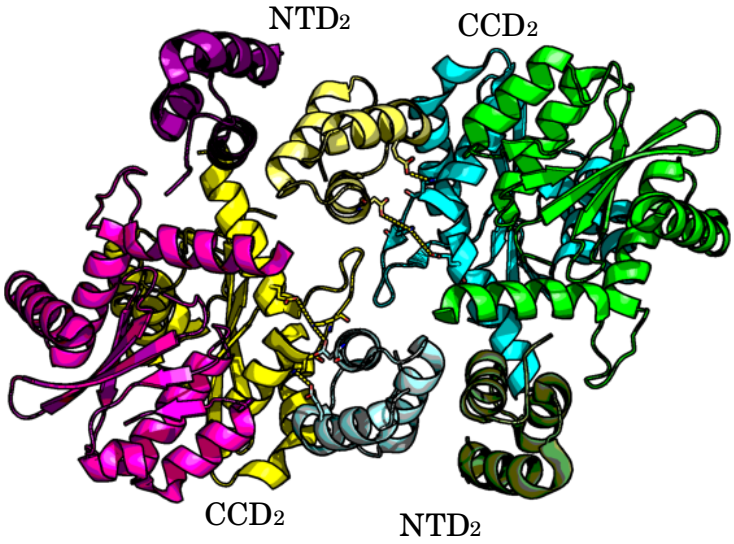
structures contiennent au moins la mutation F185K qui augmente la solubilité de la protéine tout en préservant son activité catalytique *in vitro* [82]. Le domaine catalytique se replie en une structure contenant un feuillet β à cinq brins dont le second est antiparallèle aux autres et six hélices α. Ce repliement est similaire à celui de la RNase H [7]. Une boucle flexible constitué des résidus 140 à 149 permet d'orienter le glutamate 152 en direction des deux autres résidus de la triade catalytique D64 et D116. La flexibilité de cette boucle, qualifié classiquement de boucle catalytique, complique les calculs visant à modéliser l'interaction d'un ligand avec le site catalytique (*docking* moléculaire) [18].

La particularité de la triade catalytique est sa capacité à fixer des cations métalliques divalents comme le cation magnésium (Mg^{2+}). Cependant il n'existe pas de structure cristallographique de l'intégrase du VIH-1 avec deux ions Mg^{2+} au niveau du site actif. La structure très récente de l'intégrase du PFV en interaction avec le composé GS9137 présente deux ions magnésium au niveau du site catalytique (code pdb : 3l2w) [62]. Une structure de l'intégrase du virus du sarcome aviaire (code pdb : 1vsh) [15] a aussi était déterminée en présence de deux cations Zn^{2+}. Mis à part ces deux structures, la plupart des structures cristallographiques où un cation est présent, un seul ion est fixé entre les groupes carboxylates des aspartates 64 et 116 formant un complexe de coordination avec un seul oxygène de chaque résidu et quatre molécules d'eau selon une géométrie octahédrale. Le deuxième ion pourrait être stabilisé par le squelette phosphodiester de l'ADN substrat [98].

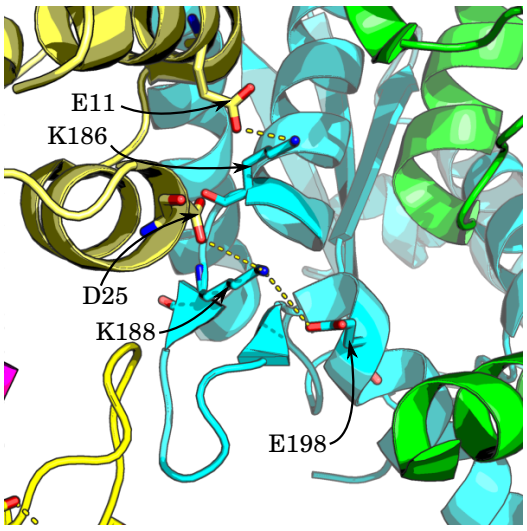
2.2.4 Organisation multimérique de l'intégrase

L'organisation multimérique de l'intégrase est nécessaire à son activité [46]. En présence d'ADN viral, l'intégrase forme un dimère à chacune de ses extrémités [51]. Les structures cristallographiques du CCD sont des structures dimériques et l'interface dimère·dimère est critique à la formation des tétramères [61]. L'observation de structures cristallographiques de l'intégrase à deux domaines (NTD-CCD) montre que la stabilisation du tétramère passe par une interaction NTD·CCD intermoléculaire (voir figure 2.3 page suivante). Ces résultats sont corroborés par l'élucidation très récente de la structure cristallographique de l'intégrase entière du PFV [62]. Pour l'intégrase du HIV-1, l'interaction intermoléculaire NTD·CTD passe par l'établissement de liaisons électrostatiques entre le glutamate 11 du NTD et la lysine 186 du CCD (voir figure 2.3b page suivante). La mutation d'un de ces résidus entraîne la dissociation de la forme tétramère [61, 13]. D'autres interactions

7. Ribonucléase H



(a)



(b)

FIGURE 2.3 – Structure cristallographique (code pdb : 1k6y [177]) de l'intégrase à deux domaine (NTD-CCD). (a) Ensemble de la structure, formant un tétramère $(NTD_2\text{-}CCD_2)_2$. (b) Détail des interactions NTD·CCD stabilisant la structure du tétramère. [61]

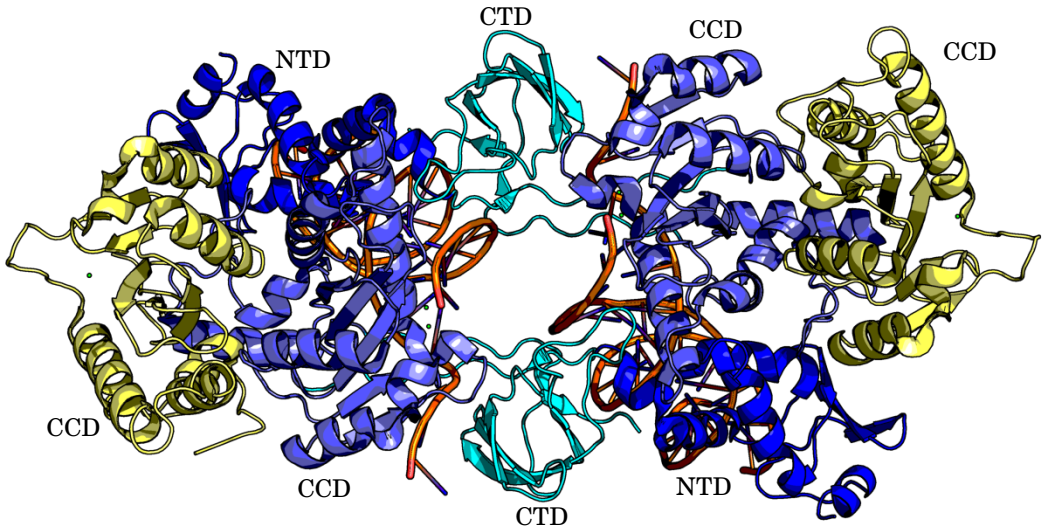


FIGURE 2.4 – Structure cristallographique (code pdb : 3l2s [62]) de l'intégrase entière du PFV en interaction avec l'ADN viral (en orange).

impliquant l'aspartate 25 du NTD, la lysine 188 et le glutamate 198 du CCD, permettent de stabiliser le complexe. Cependant ces résidus semblent moins critiques à la stabilisation du tétramère puisque leurs mutations entraînent des effets beaucoup moins drastiques que la mutation de la lysine 186 [61, 118]. Ceci est en accord avec les distances calculées entre ces différents résidus pour la structure représentée sur la figure 2.3 page ci-contre.

La figure 2.4 représente la structure cristallographique de l'intégrase entière du PFV. Sur cette figure, nous pouvons voir que le tétramère CCD_4 est stabilisé par l'interaction intermoléculaire entre le NTD d'un monomère et le dimère CCD_2, comme montré précédemment avec l'intégrase du VIH-1. Des données de modélisation par homologie montre que l'organisation des domaines de l'intégrase du HIV-1 peut adopter la même organisation globale des domaines [62]. De plus l'étude de données de microscopie électronique obtenues sur des complexes de l'intégrase du VIH-1 [123] concordent avec la structure cristallographique de l'intégrase du PFV.

2.3 Partenaires cellulaires de l'intégrase du VIH-1 : importance du LEDGF

Bien que l'intégrase recombinante seule soit nécessaire et suffisante à la réaction d'intégration *in vitro*, de nombreux travaux indiquent que le re-

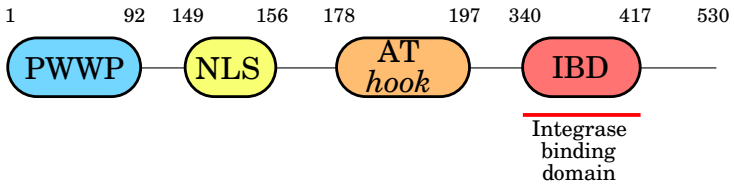


FIGURE 2.5 – Les quatre domaines fonctionnels du LEDGF.

crutement de facteurs cellulaires est nécessaire à l'intégration *in vivo*. Parmi l'ensemble des partenaires cellulaires identifiés, une attention particulière sera porté ici sur la protéine cellulaire LEDGF [8], premier cofacteur de l'intégrase responsable du ciblage et de l'intégration du génome viral dans des régions spécifiques de la chromatine [40, 24].

2.3.1 Structure du LEDGF et interaction intégrase-LEDGF

L'épissage alternatif du précurseur ARNm de LEDGF permet l'expression de deux isoformes p75 et p52 qui diffèrent seulement par leur domaine C-terminal. p52 ne possède pas de domaine de liaison à l'intégrase (IBD [9]). Dans la suite nous parlerons uniquement de l'isoforme LEDGF/p75, qui sera notée LEDGF.

Le LEDGF est une protéine de 530 acides aminés composée de quatre domaines fonctionnels (voir figure 2.5). Cette protéine possède dans sa région N-terminal un motif PWWP (de séquence proline-tryptophane-tryptophane-proline) très conservé, retrouvé dans de nombreuses protéines associées à la chromatine. Le motif AT *hook*, ou crochet AT, se trouve sous forme de deux copies entre les résidus 178-183 et 191-197. Ce domaine AT *hook* se lie aux régions de l'ADN riche en AT. La coopération des domaines PWWP et AT *hook* permet l'association du LEDGF à la chromatine [102, 170]. Un signal de localisation nucléaire (NLS [10]) est situé entre les résidus 149-156. Le LEDGF a tout d'abord était identifié comme un co-activateur général de la transcription [52].

Diverses études ont montré que le LEDGF est capable de se fixer à l'intégrase des *Lentivirus* [101, 22] et en particulier à l'intégrase du VIH-1 [24, 40]. Cette interaction stimule l'activité enzymatique d'intégration *in vitro* [101, 22] et est se fait par l'intermédiaire du domaine IBD du LEDGF,

8. *Lens Epithelial Derived Growth Factor*
9. *Integrase Binding Domain*
10. *Nuclear Localization Signal*

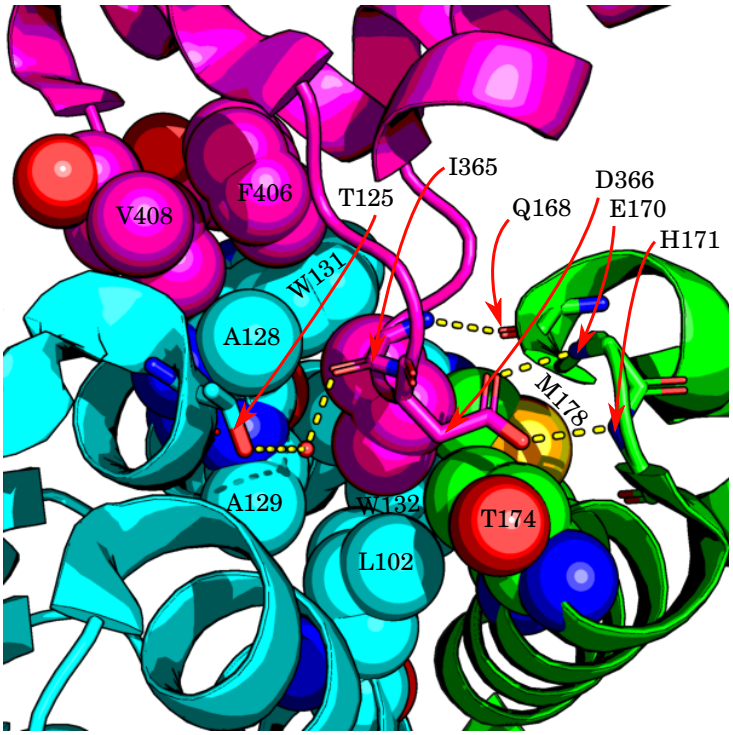


FIGURE 2.6 – Contacts intermoléculaires clés entre le dimère CCD_2 de l'intégrase du VIH-1 (en cyan et vert) et le domaine IBD du LEDGF (en magenta), pour la structure cristallographique 2b4j [23]. Les liaisons hydrogènes sont indiquées en pointillées jaunes, les acides aminés impliqués sont représentés en modèle bâton. La sphère rouge représente une molécule d'eau impliqué dans un réseau de liaisons hydrogènes. Les résidus d'acide aminé impliqués dans des contacts hydrophobes sont représentés par des sphères.

retrouvé uniquement dans la région C-terminale de l'isoforme p75 entre les résidus 340 et 417.

La structure cristallographique du complexe formé d'un dimère de domaines CCD de l'intégrase et du domaine IBD du LEDGF a été résolue [23]. Les principaux contacts mis en jeu dans la stabilisation du complexe sont représentés sur la figure 2.6. Les interactions représentées sur cette figure concordent avec les données de mutagenèse dirigée contre les résidus d'acides aminés I365, D366 et F406 du LEDGF. La mutation de ces résidus abolit l'interaction entre l'intégrase et le LEDGF. De plus la mutation de la valine 408 réduit significativement l'affinité entre ces deux protéines [25]. L'isoleucine 365 du LEDGF plonge dans une poche hydrophobe de l'intégrase formée par les résidus de la chaîne B : L102, A128, A129 et W132, et les résidus de

la chaîne A : T174 et M178. Les résidus F406 et V408 du LEDGF protège du solvant le résidu hydrophobe W131 de la chaîne B de l'intégrase. Ce résidus est exposé au solvant lorsque l'intégrase n'est pas liée au LEDGF. L'aspartate 366 du LEDGF effectue deux liaisons hydrogènes avec les amides du squelette peptidique des résidus E170 et H171 de la chaîne A de l'intégrase. L'amide de l'isoleucine 365 du LEDGF établit une liaison hydrogène avec le groupement carbonyle de la glutamine 168 de la chaîne A de l'intégrase. Une molécule d'eau se trouvant au niveau de cette interface sert de relais pour l'établissement d'une liaison hydrogène entre le carboxyle de l'isoleucine 365 du LEDGF et la fonction alcool de la thréonine 125 de la chaîne B de l'intégrase. On peut noter aussi l'existence de deux ponts salins (non représentés sur la figure 2.6 page précédente) entre la lysine 364 du LEDGF et le glutamate 170 de la chaîne A de l'intégrase, ainsi qu'entre la lysine 360 du LEDGF et l'aspartate 167 de la chaîne A de l'intégrase. La mutation de ces résidus en alanine n'entraîne pas de modification significative sur la liaison entre l'intégrase et le LEDGF [25]. Le rôle de ces résidus doit ainsi être mineur comparativement aux autres interactions décrites et représentées en figure 2.6 page précédente.

2.3.2 Rôle du LEDGF dans l'infection par le VIH-1

La capacité de l'intégrase à se lier à la chromatine est dépendante du facteur cellulaire LEDGF [101, 108]. De plus, la stabilité et l'accumulation nucléaire de l'intégrase du VIH-1 sont fortement réduites lorsque le LEDGF est sous-exprimé [100]. Ces résultats suggèrent que le LEDGF permet de lier l'intégrase à la chromatine mais aussi jouerait un rôle dans l'import nucléaire et la protection de l'intégrase contre la dégradation par le protéasome. La surexpression du domaine IBD du LEDGF dans des cellules empêche de manière drastique l'intégration du génome viral [35]. La compétition de l'IBD avec le LEDGF endogène pour le même site de fixation sur l'intégrase empêche la formation du complexe actif pouvant se lier à la chromatine.

L'interaction du LEDGF se fait exclusivement avec les intégrases des rétrovirus du genre *Lentivirus* [16]. La caractéristique des rétrovirus de ce genre par rapport aux autres genres de rétrovirus (*Alpha-*, *Beta-*, *Gamma-*, *Deltarétrovirus* et *Spumavirus*) est leur capacité à s'intégrer dans des zones transcriptionnellement active de la chromatine [157]. Cette propriété est dépendante de l'interaction intégrase·LEDGF [27].

La figure 2.7 page ci-contre propose un modèle du rôle du LEDGF dans le cycle viral du VIH-1. Dans ce modèle le domaine N-terminal (PWWP) amène

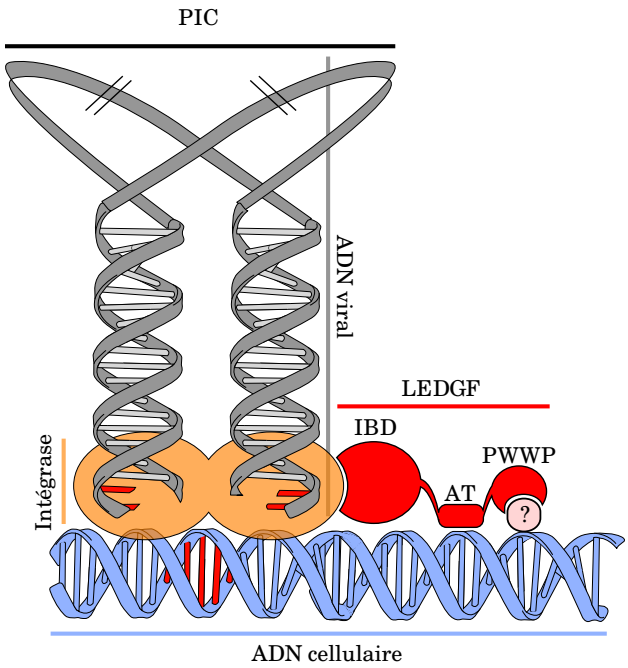


FIGURE 2.7 – Rôle du LEDGF dans le cycle viral du VIH-1. Le LEDGF interagit avec l'intégrase du complexe de pré-intégration (PIC : *Pre-Integration Complex*) par le domaine C-terminal IBD. Le domaine AT, contenant les deux motifs AT *hook*, interagit avec l'ADN cellulaire cible. Le domaine PWWP permet le ciblage du PIC au sein d'une zone transcriptionnellement active de la chromatine par son interaction avec un facteur inconnu de la chromatine (noté " ?"). D'après Hare & Cherepanov [60].

le complexe de pré-intégration (PIC [11]) dans une zone transcriptionnellement active de la chromatine en se fixant à un facteur chromatinien. Le domaine AT stabilise cette interaction en s'arrimant à l'ADN cible cellulaire. Le LEDGF est relié au PIC via l'intégrase par le domaine IBD du LEDGF.

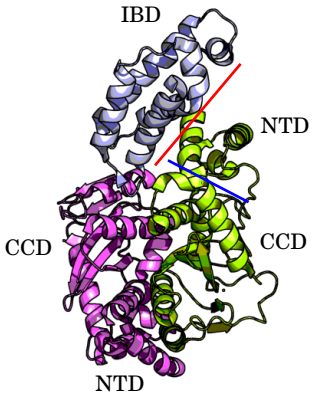
2.3.3 Rôle du LEDGF dans la tétramérisation de l'intégrase

Nous avons vu dans le paragraphe 2.2.4 page 25 que l'organisation multimérique de l'intégrase est nécessaire à son activité. De plus nous avons vu que le domaine N-terminal de l'intégrase était important à la formation de tétramère.

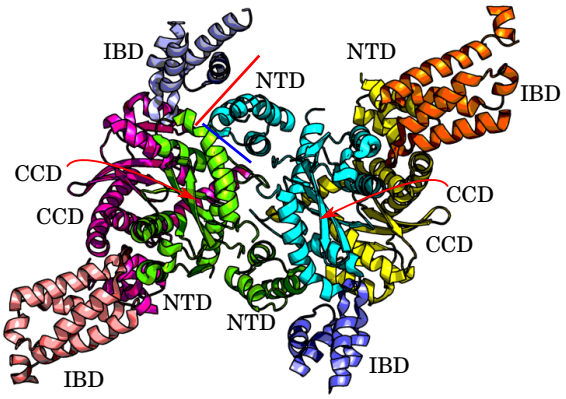
L'élucidation récente de la structure cristallographique d'un fragment à deux domaines (NTD + CCD) de l'intégrase du VIH-2 en interaction avec l'IBD du LEDGF (code pdb : 3f9k [63]), a montré que le complexe intégrase·LEDGF est stabilisé par une interaction entre le NTD et l'IBD – représentée par la ligne rouge sur la figure 2.8a page ci-contre. L'interaction NTD·CCD décrite précédemment est représentée en bleu sur cette même figure. Ces données sont en accords avec des résultats antérieurs montrant que le NTD est nécessaire à une interaction de haute affinité avec le LEDGF [108]. Cependant du fait de l'organisation dimérique de l'intégrase dans cette structure, la compréhension du mécanisme moléculaire de stabilisation du tétramère d'intégrase par le LEDGF reste incomplète.

La résolution de la structure de l'intégrase contenant les domaines NTD-CCD du MVV [12] en complexe avec l'IBD du LEDGF a permis de comprendre les interactions mis en jeu dans la stabilisation du tétramère d'intégrase par le LEDGF (code pdb : 3hph) [61]. La figure 2.8b page suivante donne la structure de l'ensemble du complexe $((\text{CCD-NTD})_2{\cdot}\text{IBD}_2)_2$ et un modèle simplifié de l'architecture globale de ce complexe est représenté en figure 2.8c page ci-contre. Dans ce modèle le domaine N-terminal de l'intégrase joue un rôle cruciale dans la formation du tétramère actif de l'intégrase. En effet celui ci stabilise le tétramère en établissant une interaction avec le CCD du dimère opposé. Le domaine IBD du LEDGF stabilise le tétramère en se liant à la fois aux CCD d'un dimère et au NTD du dimère opposé. Il a été démontré que le LEDGF stimule la tétramérisation de l'intégrase *in vitro* [118]. Afin d'expliquer ce mécanisme, un modèle allostérique a été développé. La fixation de l'IBD du LEDGF sur le dimère CCD_2 pourrait entraîner, par un mécanisme allostérique, la formation de l'interface CCD/NTD [66].

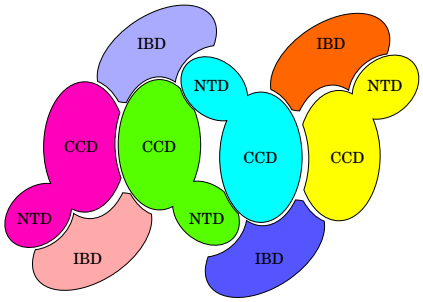
11. *Pre-integration Complex*
12. Maedi-Visna Virus



(a) Interactions CCD-NTD·IBD intra-dimères de l'intégrase du VIH-2 (code pdb : 3f9k). La ligne bleu indique l'interaction intra-dimère NTD·CCD et la ligne rouge indique l'interaction NTD·IBD.

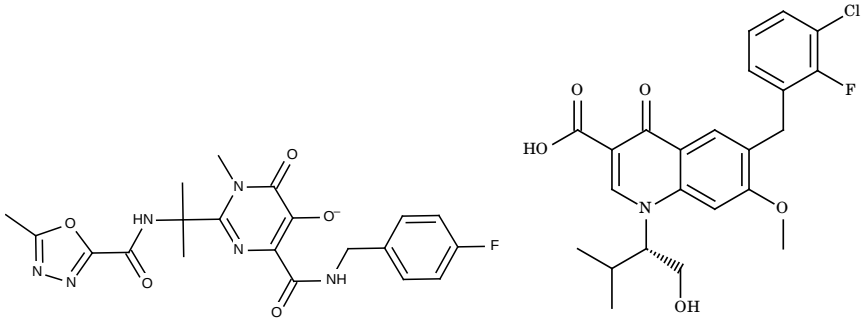


(b) Interaction CCD-NTD·IBD intra-dimères et inter-dimères de l'intégrase du MVV (code pdb : 3hph). La ligne bleu indique l'interaction inter-dimère NTD·CCD et la ligne rouge indique l'interaction NTD·IBD.



(c) Représentation simplifiée du complexe présenté en figure b.

FIGURE 2.8 – Interactions entre les domaines NTD, CCD de l'intégrase des lentivirus et le domaine IBD du LEDGF. Implication du LEDGF dans la multimérisation de l'intégrase.



(a) MK0518 (b) GS9137

FIGURE 2.9 – Inhibiteurs "catalytiques" de l'intégrase.

2.4 Inhibition de l'intégrase du VIH-1

2.4.1 Inhibiteurs "catalytiques"

Sept années de recherche, depuis la découverte des premiers inhibiteurs de l'intégrase en 1993, ont été nécessaire pour le développement des premiers inhibiteurs sélectifs du transfert de brin. Le raltegravir (MK0518, figure 2.9a) a été approuvé par la FDA le 12 octobre 2007 comme inhibiteur sélectif du transfert de brin et ouvre la voie à une nouvelle famille de médicaments anti-VIH ciblant la troisième enzyme de ce virus. L'elvitegravir (GS9137, figure 2.9b) est lui en phase III d'essai clinique.

Le mode d'action des inhibiteurs "catalytiques" sélectifs du transfert de brin est schématisé sur la figure 2.10 page suivante et est comparé au mode d'action des inhibiteurs non sélectif du transfert de brin, bloquant aussi le *3'-processing* [85].

En principe, l'inhibition de l'intégrase peut avoir lieu à deux moments clés du processus :

- avant l'interaction avec l'ADN viral. On parle alors d'inhibiteur du *3'-processing*. Ces inhibiteurs ne se lient alors qu'à un seul cation métallique Mg^{2+}.
- avant l'interaction avec l'ADN de l'hôte. On parle alors d'inhibiteur du transfert de brin. Ces inhibiteurs se lient aux deux cations métalliques Mg^{2+}.

Les inhibiteurs du transfert de brin appartiennent en général à la famille des dicétoacides et dérivés. C'est dans cette famille qu'ont été identifiées les molécules présentant les meilleurs activités antirétrovirales. Les polyphénols inhibent souvent l'étape de *3'-processing*, mais de manière non sélective – ils

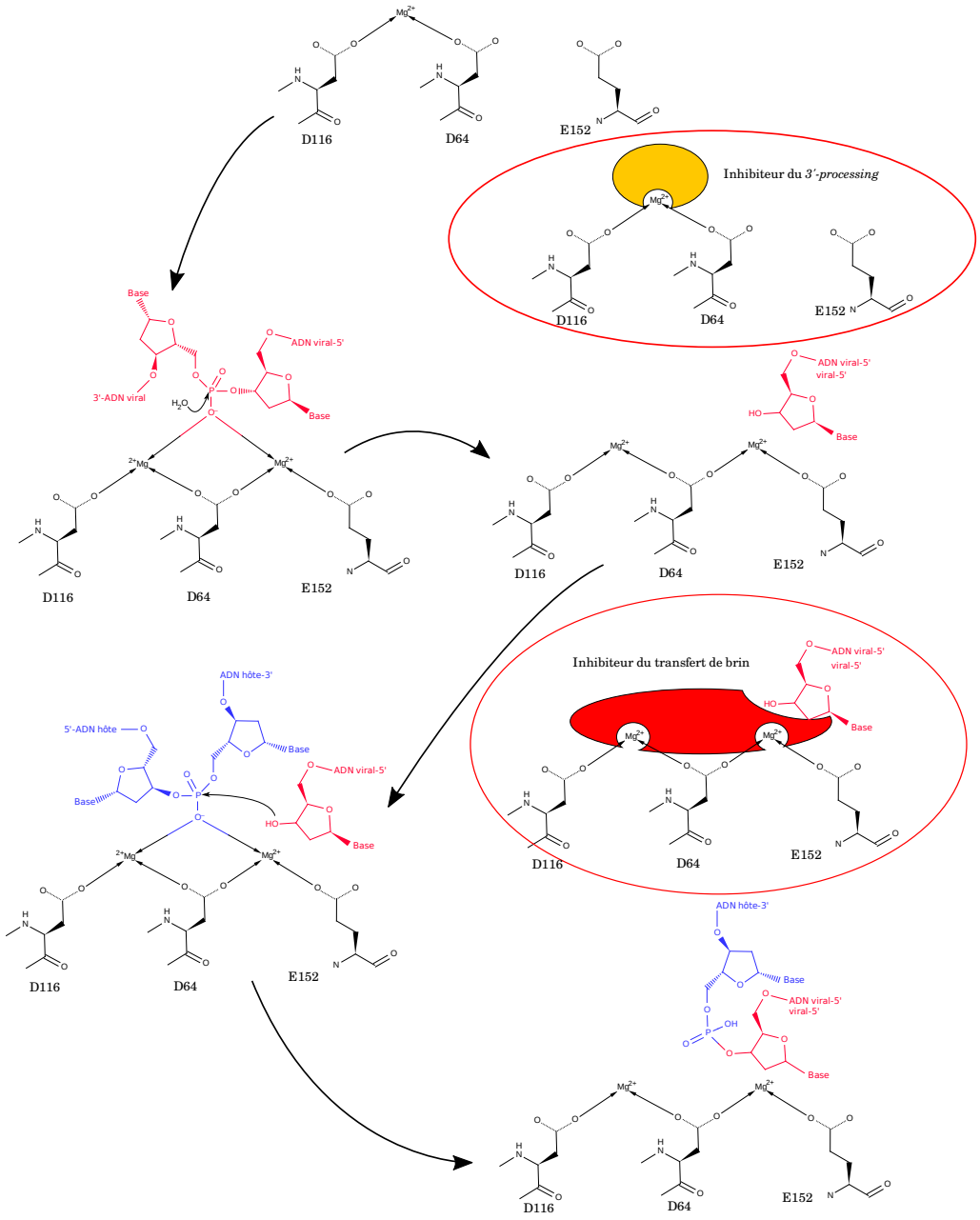
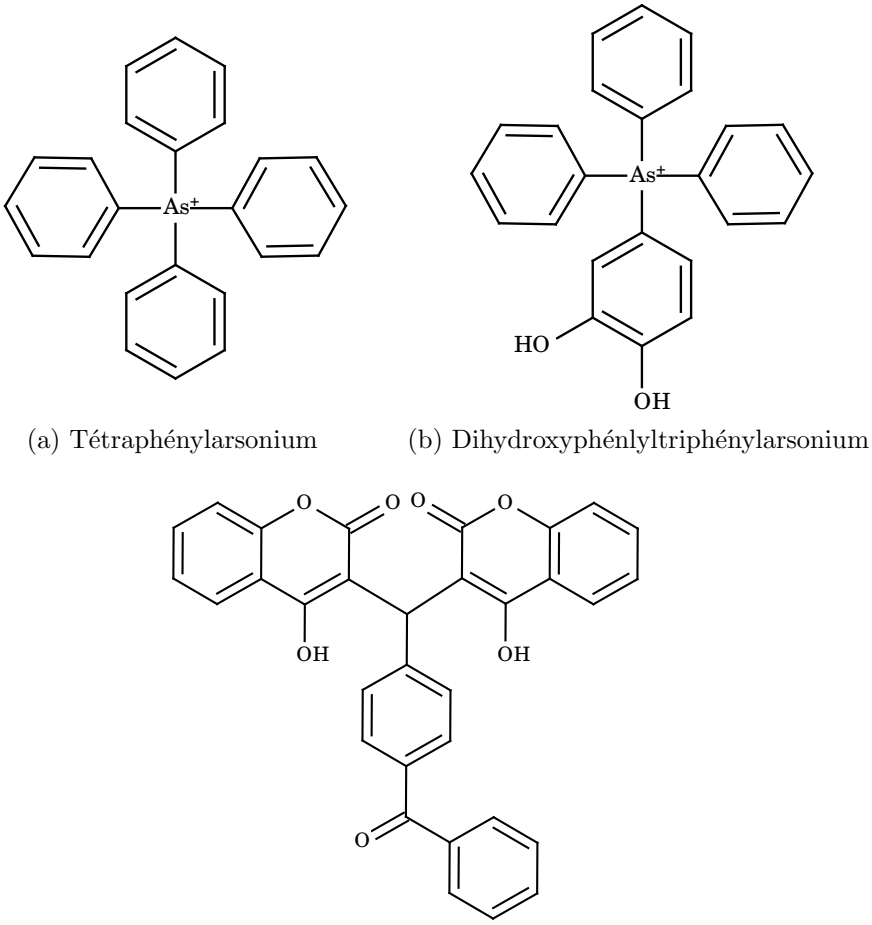


FIGURE 2.10 – Mécanisme d'action des inhibiteurs de l'intégrase (zone cerclée de rouge) dans son cycle catalytique.



(a) Tétraphénylarsonium (b) Dihydroxyphénlyltriphénylarsonium

(c) Dérivé de coumarine

FIGURE 2.11 – Inhibiteurs "non catalytiques" de l'intégrase.

inhibent aussi le transfert de brin – et non spécifique.

2.4.2 Inhibiteurs "non catalytiques"

Nous allons focaliser notre étude sur l'identification de petites molécules bloquant l'interaction entre l'intégrase et le LEDGF. Avant la mise en évidence du LEDGF comme partenaire d'interaction de l'intégrase, deux molécules : le tétraphénylarsonium (voir figure 2.11a) et le dihydroxyphénlyltriphénylarsonium (voir figure 2.11b), ont été identifiées comme des inhibiteurs de l'intégrase, bien que seulement le deuxième inhibe faiblement l'activité enzymatique *in vitro* [126].

Le site de fixation identifié coïncide avec le sillon, présent sur le dimère CCD_2 de l'intégrase, accueillant la boucle de l'IBD du LEDGF (voir figure 2.6 page 29). Le tétraphénylarsonium effectue une interaction charge-charge avec la fonction carbonyle de la glutamine 168 de l'intégrase, tandis qu'un cycle aromatique de ce composé interagit avec les tryptophanes 131 et 132. Un dérivé de coumarine (voir figure 2.11c page précédente) [2] a aussi été découvert comme interagissant dans une région proche de la région précédemment décrite. Des dérivés de coumarines présentent des IC_{50} comprises entre 7 et 20 μM sur l'intégrase, *in vitro*.

Des études plus récentes ont montré que le peptide contenant les résidus compris entre les positions 355 et 377 de la séquence du LEDGF peut influer sur l'interaction entre l'intégrase et le LEDGF avec une IC_{50} de 25μM [1].

Deuxième partie

Approche méthodologique

Chapitre 3

Modélisation moléculaire

The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble.

Paul Dirac

3.1 *Docking*

3.1.1 Introduction

Le *docking*, ou arrimage moléculaire, permet de prédire la conformation d'une molécule ligand au sein d'une macromolécule cible. Le logiciel de *docking* permet d'optimiser les interactions ligand·récepteur par une exploration systématique de la conformation du ligand au sein du site de fixation. Il est généralement divisé en deux parties :

- l'algorithme de *docking* à proprement parlé, qui permet l'exploration de l'espace conformationnel,
- la fonction de score qui permet d'évaluer l'énergie d'interaction entre la molécule dockée et le récepteur.

Le *docking*, du fait de la difficulté de prévoir une énergie d'interaction à partir d'une structure tridimensionnelle, produit un ensemble de résultats qui correspond à un ensemble de conformations ligand·protéine. La technique calculatoire du *docking* peut être utilisée pour répondre à deux types de problème :

- nous connaissons une molécule active, agissant sur une cible donnée, mais la structure tridimensionnelle du complexe ligand·récepteur est

inconnue. Dans ce cas le *docking* permet de prévoir une structure tridimensionnelle hypothétique du complexe protéine·ligand. La fonction de score permet alors de classer les différentes poses du même ligand au sein du récepteur. En général, en l'absence de données expérimentales, la pose retenue est la pose de plus basse énergie ;
- nous cherchons à découvrir des molécules actives dans une chimiothèque pour une cible considérée. Dans ce cas, après *docking* de l'ensemble des molécules de la base de donnée, celles-ci sont classés par énergie d'interaction croissante, c'est à dire des complexes les plus stables au complexes les moins stables, selon les prédictions de la fonction de score. Cette technique correspond alors à un calcul de criblage virtuel basé sur la structure du récepteur ou *structure-based virtual screening*.

La réussite d'un *docking* repose alors sur deux conditions :
- la capacité de l'algorithme générateur de conformations à produire au moins une conformation exacte parmi les n poses produites
- la capacité de la fonction de score à classer la bonne conformation comme la meilleure conformation.

La fonction de score est donc primordiale dans le processus de *docking*. Il en existe trois types :
- fonction de score explicite dépendante d'un champ de force (AMBER ou CHARMM) (*explicit force field scoring function*)
- fonction de score empirique (*empirical scoring function*) donnant l'enthalpie libre selon la formule :

$$\Delta G = \sum_{i \in interactions} \Delta G_i \cdot f_i(l, r) \tag{3.1}$$

où l et r sont les termes traduisant la géométrie du ligand
- fonction de score basée sur des données statistiques obtenues à partir de structures obtenues expérimentalement (*knowledge based scoring function*).

Trois logiciels de *docking* ont été utilisés : AutoDock [127], DOCK [94, 44, 128, 96] et Surflex-Dock [79].

3.1.2 AutoDock

Exploration de l'espace conformationnel

Le logiciel AutoDock [127] implémente trois algorithmes d'exploration de l'espace conformationnel.
- Un recuit simulé utilisant la méthode Monte Carlo [122]
- Un algorithme génétique [70]

– Un algorithme génétique Lamarckien [127]

Nous allons décrire uniquement les algorithmes génétiques du fait de la limitation de la méthode du recuit simulé à un petit nombre de degrés de liberté du ligand (moins de huit liaisons de libre rotation).

Les algorithmes génétiques utilisent des variables d'état décrivant les translations, rotations et conformations possibles du ligand au sein du site de *docking*. Chaque variable d'état correspond à un gène. L'ensemble des variables d'état décrivant une pose de *docking* représente donc un génotype. Les coordonnées atomiques découlant d'un génotype correspondent au phénotype. Les phénotypes peuvent être évalués grâce à la fonction de score qui lui associe ainsi une énergie d'interaction. L'algorithme est initialisé aléatoirement. Des paires aléatoires d'individus sont ainsi croisées en utilisant un processus de *crossing-over*. Chaque descendant reçoit des gènes provenant de l'ensemble de ses parents, et chaque gène peut muter de manière aléatoire. Les individus autorisés à se reproduire sont ensuite sélectionnés par une évaluation d'énergie par la fonction de score ; les complexes ligand·récepteur les plus stables sont ainsi sélectionnés à chaque génération. L'algorithme se termine après un nombre donné d'itération (nombre de sélections positives ou négatives lors des itérations).

L'algorithme génétique Lamarckien repose sur le même principe que l'algorithme génétique standard décrit ci-dessus. Jean-Baptiste de Lamarck, naturaliste français, a proposé la théorie de la transmission des caractères acquis. Le caractère acquis pour l'algorithme de *docking* envisagé provient d'une minimisation énergétique avant la sélection. La génération suivante provient ainsi de structures minimisées énergétiquement (figure 3.1 page suivante). L'algorithme génétique Lamarckien est plus rapide que l'algorithme génétique standard dans la recherche du minimum.

Fonction de score

La fonction de score AutoDock [127] est une fonction de score semi-empirique comportant des coefficients provenant d'analyses statistiques de complexes protéine·ligand avec des constantes d'association connues. Cette

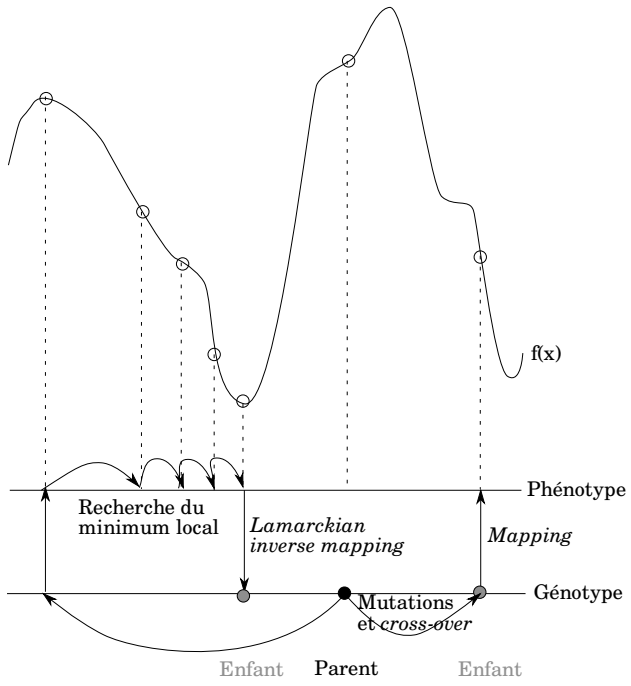


FIGURE 3.1 – Comparaison de l'algorithme génétique standard et de l'algorithme génétique Lamarckien.

fonction comporte cinq termes :

$$\Delta G = \Delta G_{VdW} \sum_{i,j} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \right) \quad (3.2)$$

$$+\Delta G_{liaisonsH} \sum_{i,j} E(t) \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right) \quad (3.3)$$

$$+\Delta G_{elec} \sum_{i,j} \frac{q_i q_j}{\epsilon(r_{ij}) r_{ij}} \quad (3.4)$$

$$+\Delta G_{tor} N_{tor} \quad (3.5)$$

$$+\Delta G_{sol} \sum_{i,j} (S_i V_j + S_j V_i) e^{(-r_{ij}^2/2\sigma^2)} \quad (3.6)$$

Les cinq termes en ΔG sont des coefficients déterminés empiriquement à partir d'une analyse par régression linéaire à partir de complexes protéine·ligand avec des constantes d'association connues. La somme sur i et j est réalisée sur l'ensemble des paires d'atomes appartenant au ligand et au récepteur, respectivement. Les trois premiers termes, classiques de la mécanique moléculaire, décrivent les interactions de van der Waals, les liaisons hydrogènes et les interactions électrostatiques. Les deux derniers termes apportent le facteur entropique lié à la diminution de l'espace conformationnel et à la désolvatation induites par la fixation du ligand sur le récepteur.

Les paramètres A_{ij} et B_{ij}, décrivant les interactions de van der Waals, sont basés sur le champ de force Amber [176].

Les paramètres C_{ij} et D_{ij} sont calibrés de telle sorte que les valeurs d'énergie de liaison hydrogène (terme 3.3) soit de $5\,\mathrm{kcal} \cdot \mathrm{mol}^{-1}$ pour une liaison hydrogène idéale avec une distance de 1,9 Å entre un atome d'azote et un atome d'oxygène et de $1\,\mathrm{kcal} \cdot \mathrm{mol}^{-1}$ pour une liaison hydrogène impliquant un atome de soufre et pour une distance de 2,5 Å. Le facteur $\Delta G_{liaisonsH}$ module l'énergie calculée en apportant la contribution géométrique basée sur l'angle entre donneur et accepteur de liaison hydrogène.

L'énergie de Coulomb (terme 3.4) est basée sur la constante diélectrique de Mehler et Solmajer [120] dépendante de la distance inter-atomique ($\epsilon(r_{ij})$).

Le terme de désolvatation (terme 3.6) dépend des volumes des atomes considérés (V) pondérés par un paramètre de solvatation (S). Une fonction exponentielle dépendante de la distance inter-atomique (r_{ij}) est pondérée par la constante σ fixée à 3,5Å.

Le calcul de l'énergie d'interaction est basé sur une grille (générée par AutoGrid) représentant le récepteur. Cette méthode permet de simplifier le calcul : à chaque point de la grille la valeur d'énergie est calculée en se basant sur le type d'atome, les charges et la position au sein de la grille. La grille

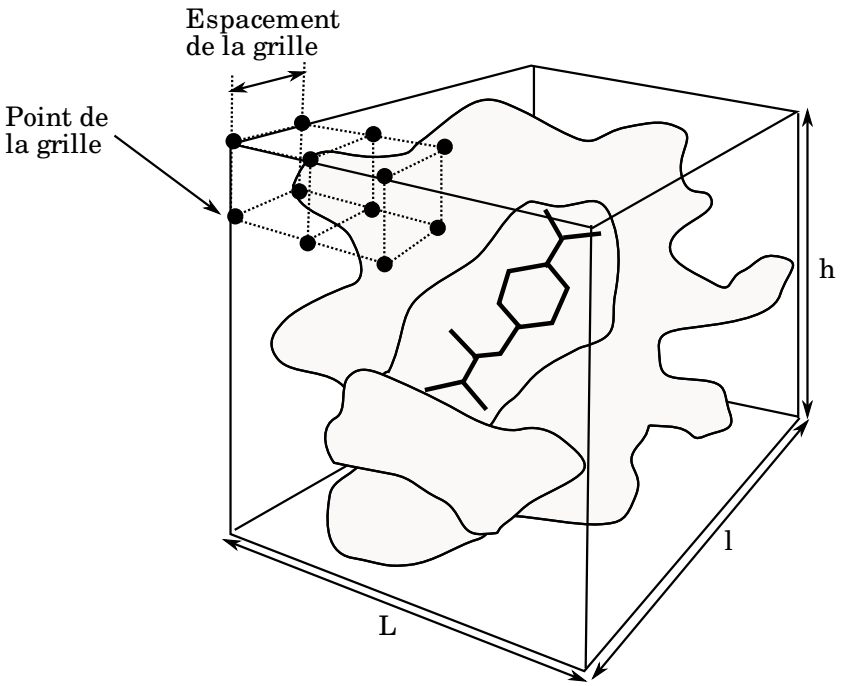


FIGURE 3.2 – Paramètres descriptifs de la grille générée par AutoGrid

est décrite par un ensemble de paramètres à savoir les dimensions de la grille (longueur et largeur du parallélépipède) et l'espacement des points (maillage de la grille) (figure 3.2).

3.1.3 Dock

Dock 1.0 est le premier logiciel de *docking* recepteur·ligand automatique développé. Il a été décrit pour la première fois en 1982 [94]. Il a été développé par le département de Pharmacologie de l'université de Californie San-Francisco (UCSF).

Exploration de l'espace conformationnel

Dock est basé sur la complémentarité stérique entre ligand et récepteur. La surface moléculaire est générée de manière classique par l'utilisation d'une sphère roulant sur les sphères de van der Waals des différents atomes (figure 3.3 page suivante). Une image négative de la poche de *docking* est alors représentée par un ensemble de sphères de diamètre différent tangentes à la surface moléculaire en exactement deux points (figure 3.4 page ci-contre). Les atomes du ligand sont ensuite superposés sur les centres des sphères ainsi

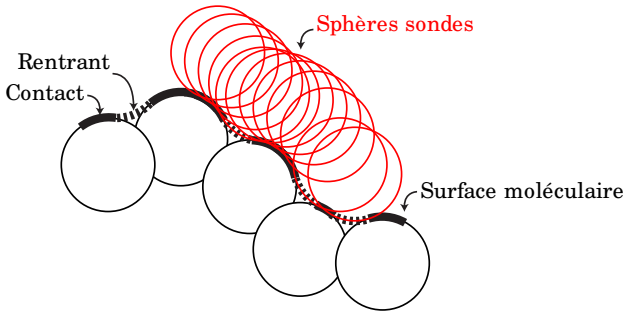


FIGURE 3.3 – La surface moléculaire est générée par une sphère sonde de 1,4Å de rayon (afin de représenter une molécule d'eau) roulant sur les sphères de van der Waals des différents atomes. La surface moléculaire correspond aux points de contact entre la sonde et les sphères de van der Waals ainsi que les rentrants.

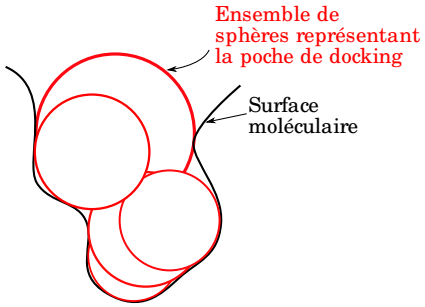


FIGURE 3.4 – Représentation de la poche de *docking* par un ensemble de sphères.

générées. Une fois l'orientation générée l'interaction ligand·récepteur est optimisée énergétiquement. Pendant la minimisation le ligand est autorisé à être flexible, le récepteur, lui, restant rigide. Afin de prendre en compte la flexibilité du ligand, le ligand est coupé en fragments rigides. Chaque fragment est docké comme indiqué précédemment. Ensuite les fragments rigides sont reliés par les fragments flexibles. Chaque orientation ainsi générée est évaluée énergétiquement par la fonction de score. Cet algorithme est qualifié d'algorithme *anchor and grow* [44].

Fonction de score

La fonction de score DOCK est une fonction de score explicite dépendante du champ de force AMBER. La fonction de score est dépendante de deux termes :

- un terme potentiel de Lennard-Jones (interactions de van der Waals)
- un terme électrostatique

$$E = \sum_{i}^{lig} \sum_{j}^{rec} \left(\frac{A_{ij}}{r_{ij}^{a}} - \frac{B_{ij}}{r_{ij}^{b}} + 332 \frac{q_i q_j}{\epsilon(r_{ij}) r_{ij}} \right) \tag{3.7}$$

3.1.4 Surflex-Dock

Exploration de l'espace conformationnel

Le programme Surflex-Dock utilise un *protomol* [78] afin de décrire le site de *docking*. Le *protomol* est construit à partir de trois sondes différentes :

- $C = O$ représentant les accepteurs de liaison hydrogène
- NH représentant les donneurs de liaison hydrogène
- CH_4 représentant les zones hydrophobes

La molécule à docker est fragmentée en parties rigides. Chaque fragment rigide est aligné sur le *protomol* en utilisant une fonction Gaussienne dépendante de la distance entre les surfaces complémentaires du *protomol* et du fragment docké [77]. L'optimisation de l'alignement utilise une représentation sous forme de triangle. Des triangles de tailles identiques représentent une complémentarité chimique. Une transformation qui donne lieu à une bonne complémentarité chimique donne naissance à une pose de haut score.

Le fragment associé au meilleur score de *docking* est utilisé comme tête (*head*) pour la reconstruction de la molécule entière. La pose est optimisée localement pour s'adapter au mieux au récepteur. Le fragment suivant (queue ou *tail*) est alors aligné avec le *protomol* mais est soumis à des contraintes afin que la queue puisse s'associer à la tête.

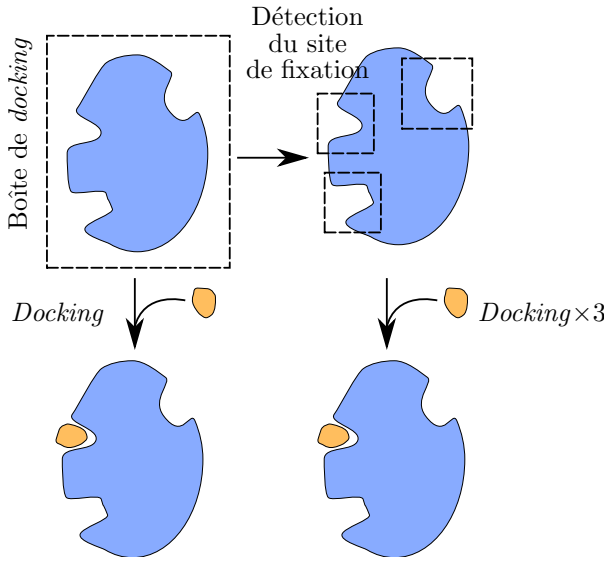


FIGURE 3.5 – Les deux stratégies envisageables lors d'un *blind docking*.

Fonction de score

La procédure de *scoring* du logiciel Surflex-Dock repose sur une fonction de score empirique basée sur l'analyse statistique de 34 structures de complexe protéine·ligand (16 protéines différentes) allant de 10^{-3} à 10^{-14} M en terme de K_d. Elle résulte de la combinaison linéaire de quatre fonctions non-linéaires dépendantes des distances entre la surface du récepteur et du ligand. Les quatre termes considérés sont les suivants :

- terme stérique
- terme polaire
- terme entropique caractérisant les degrés de liberté
- terme entropique de solvatation

3.1.5 *Docking* à l'aveugle ou *blind docking*

Lorsque le site de fixation du ligand étudié n'est pas connu, il est possible d'effectuer un *docking* sur l'ensemble de la surface moléculaire : on parle alors de *docking* à l'aveugle ou encore de *blind docking* [67]. Deux stratégies sont envisageables afin de réaliser un *blind docking* (voir figure 3.5).

La première consiste à créer une boîte englobant l'ensemble de la surface moléculaire du récepteur [67]. Le calcul de *docking* est alors réalisé sur l'ensemble de la protéine ce qui conduit à deux limitations :

- le temps de calcul est long

– la maille de la grille de *docking* doit être assez grande afin de pouvoir englober l'ensemble de la surface moléculaire et que le temps de calcul reste raisonnable.

La deuxième approche consiste à prédire des sites potentiels de fixation des ligands [55]. Cette approche repose sur le calcul des MIFs – *Molecular Interaction Fields* [54]. Les MIFs décrivent les variations spatiales de l'énergie d'interaction entre la cible considérée et un atome, ou un groupement chimique, sonde. Le calcul est discrétisé par l'utilisation d'une grille tridimensionnelle entourant l'ensemble de la protéine cible. Le calcul des MIFs est réalisé par le logiciel EasyMIF [54] qui utilise le champ de force GROMOS [171] et une constante diélectrique dépendante de la distance. Le logiciel SiteHound [54] est ensuite utilisé afin de sélectionner les zones d'interaction favorables et de les regrouper afin de représenter la totalité d'un site de fixation. Les différents sites ainsi trouvés sont ensuite classés selon le potentiel d'interaction.

3.1.6 *Post-processing*

Le *post-processing* correspond à un ensemble de calculs supplémentaires effectués après un *docking* afin d'affiner et/ou de reclasser un ensemble de poses de *docking* d'une même molécule ou de molécules différentes. Un résultat de *docking* est composé des coordonnées cartésiennes du ligand dans le repère du récepteur ainsi que des estimations d'énergies de liaison souvent décomposées suivant les différents termes de la fonction de score. Les algorithmes utilisables en *post-processing* utilisent ces données en entrée. Le choix des outils de *post-processing* est important, et conditionne souvent les résultats obtenus lors d'un criblage virtuel (voir section 3.2 page 55) [107].

Une telle analyse des résultats de *docking* peut être menée grâce à l'utilisation de multiples protocoles :

– calcul de l'enfouissement du ligand [162]
– prise en compte de l'entropie des chaînes latérales de la protéine [56]
– combinaison du *docking* avec des méthodes QSAR [1] [88]
– affinement des poses de *docking* par MM–PBSA [2] ou MM–GBSA [3] [137] (voir page 53)
– utilisation d'une fonction de score topologique [111]
– utilisation d'une carte auto-organisatrice (voir chapitre 4 page 67) [14]

L'affinement des poses de *docking* peut aussi permettre l'introduction de la flexibilité du récepteur. Cette flexibilité peut être introduite par *do-*

1. *Quantitative Structure-Activity Relationship*
2. *Molecular Mechanics – Poisson-Boltzmann Surface Area*
3. *Molecular Mechanics – Generalized Born Surface Area*

cking sur de multiples conformations du récepteur [4] [167], ou par mécanique moléculaire avec flexibilité partielle ou totale du récepteur [57].

Le *post-processing* consiste le plus souvent en une nouvelle prédiction des énergies d'interaction à partir des poses générées.

CScore

Les fonctions de score sont pour la plupart empiriques et découlent d'un nombre restreint de données expérimentales. Elles essayent d'être le plus général possible et de s'appliquer au maximum de problèmes possibles. Cependant du fait de leurs différences vis à vis de leur calibration, le *scoring* est dépendant de la cible, tout comme l'efficacité de l'algorithme de docking. CScore regroupe un ensemble de fonctions de score [29] permettant ainsi le choix de la fonction de score la mieux adapté au problème et même l'établissement d'une fonction de score consensus provenant de l'ensemble des résultats de *scoring*. CScore regroupe quatre fonctions de score : G_score [84], D_score [95], PMF_score [130] et ChemScore [39].

G_Score : Cette fonction de score est composée de trois termes :

- énergie du complexe ligand·protéine :

$$E_{complexe} = \sum_{i}^{lig} \sum_{j}^{prot} \left(\frac{A}{r_{ij}^{8}} - \frac{B}{r_{ij}^{4}} \right) \tag{3.8}$$

  où r_{ij} sont des distances inter-atomiques
- énergie interne (intra-ligand) :

$$E_{int} = E_{vdw} + E_{tors} \tag{3.9}$$

  avec E_{tors}, l'énergie de torsion du ligand et E_{vdw} donnée par la relation :

$$E_{vdw} = \sum_{i}^{lig} \sum_{j}^{prot} \left(\frac{C}{r_{ij}^{12}} - \frac{D}{r_{ij}^{6}} \right) \tag{3.10}$$

- énergie de liaison hydrogène, basée sur le type des atomes (donneurs et accepteurs de liaison hydrogène) et la géométrie du système (incluant les interactions avec des métaux).

4. Cette technique nécessite un nouveau *docking* des molécules sélectionnées, cependant elle est classée dans ce paragraphe intitulé *post-processing* car elle implique une pré-sélection afin de rester dans des temps de calculs raisonnable lors d'un criblage virtuel (section 3.2 page 55).

PMF_Score : Cette fonction de score est basée sur l'analyse d'un grand nombre de complexes protéine·ligand extrait de la *Protein Data Bank* (PDB). Est ainsi extrait une fonction décrivant l'énergie libre d'Helmholtz des interactions protéine·ligand (Potential of Mean Force, PMF) :

$$PMF_{score} = \sum_{i}^{lig} \sum_{j}^{prot} A_{ij} r_{ij} \tag{3.11}$$

où $A_{ij}r_{ij}$ est la contribution d'une paire d'atome de type i et de type j à une distance r_{ij}.

D_Score : Cette fonction de score est basée uniquement sur les interactions entre charges et de van der Waals entre les atomes de la protéine et du ligand :

$$D_{score} = \sum_{i}^{lig} \sum_{j}^{prot} \left(\frac{A}{r_{ij}^{12}} - \frac{B}{r_{ij}^{6}} + 332 \frac{q_i q_j}{D r_{ij}} \right) \tag{3.12}$$

avec q_i la charge de l'atome i et D la constante diélectrique.

ChemScore : Trois termes composent cette fonction :

– énergie de liaison hydrogène :

$$\Delta G_{liaisonH} = \sum_{i}^{lig} \sum_{j}^{prot} g_1(\Delta r_{ij}) g_2(\Delta \alpha_{ij}) \tag{3.13}$$

avec :

$$g_1(\Delta r_{ij}) = \begin{cases} 1 & \text{si } \Delta r_{ij} \leq 0,25\text{Å} \\ 1 - \frac{\Delta r_{ij} - 0,25}{0,4} & \text{si } 0,25\text{Å} < \Delta r_{ij} \leq 0,65\text{Å} \\ 0 & \text{si } \Delta r_{ij} > 0,65\text{Å} \end{cases} \tag{3.14}$$

et,

$$g_2(\Delta \alpha_{ij}) = \begin{cases} 1 & \text{si } \Delta \alpha_{ij} \leq 30^\circ \\ 1 - \frac{\Delta \alpha_{ij} - 30}{50} & \text{si } 30^\circ < \Delta \alpha_{ij} \leq 80^\circ \\ 0 & \text{si } \Delta \alpha_{ij} > 80^\circ \end{cases} \tag{3.15}$$

où, Δr_{ij} est la déviation par rapport à une longueur de liaison H$\cdots$O/N idéale de $1,85$Å et $\Delta\alpha$ la déviation par rapport à un angle idéal de 180° pour une liaison hydrogène N/O-H$\cdots$O/N.

– énergie de coordination métal-ligand :

$$\Delta G_{metal} = \sum_{i}^{lig} \sum_{M}^{prot} f(\Gamma_{iM}) \tag{3.16}$$

avec Γ_{iM} la distance entre l'atome i du ligand et le métal M de la protéine et

$$f(\Gamma_{iM}) = \begin{cases} 1 & \text{si } \Gamma_{iM} \leq 2,2\text{Å} \\ 1 - \frac{\Gamma_{iM}-2,2}{0,4} & \text{si } 2,2\text{Å} < \Gamma_{iM} \leq 2,6\text{Å} \\ 0 & \text{si } \Gamma_{iM} > 2,6\text{Å} \end{cases} \tag{3.17}$$

– énergie de contact hydrophobe :

$$\Delta G_{lipo} = \sum_{i}^{lig} \sum_{j}^{prot} f(\Gamma_{ij}) \tag{3.18}$$

avec Γ_{ij} la distance entre l'atome lipophile i du ligand et l'atome lipophile j de la protéine.

$$f(\Gamma_{ij}) = \begin{cases} 1 & \text{si } \Gamma_{ij} \leq R_1\text{Å} \\ 1 - \frac{\Gamma_{ij}-R_1}{3,0} & \text{si } R_1\text{Å} < \Gamma_{ij} \leq (R_1+3,0)\text{Å} \\ 0 & \text{si } \Gamma_{ij} > (R_1+3,0)\text{Å} \end{cases} \tag{3.19}$$

où

$$R_1 = r_{vdW}(i) + r_{vdW}(j) + 0,5 \tag{3.20}$$

avec $r_{vdW}(i)$ le rayon de van der Waals de l'atome i.

– entropie rotationnelle, qui représente la diminution de la liberté conformationnelle du ligand suite à sa fixation sur la protéine :

$$\Delta G_{rot} = 1 + \left(1 - \frac{1}{N_{rot}}\right) \sum_{i}^{N_{rot}} \frac{P_{nl}(i) + P'_{nl}(i)}{2} \tag{3.21}$$

avec N_{rot} le nombre de liaisons de libre rotation figées au sein du récepteur et $P_{nl}(i)$ et $P'_{nl}(i)$ la proportion d'atomes non hydrophobes de part et d'autre de la liaison de libre rotation i.

MM–GBSA

Cette technique permet de calculer une énergie d'interaction entre un récepteur et son ligand. L'implémentation de cette méthode au sein du logiciel Dock 6.4 [96] a été utilisée. Cette implémentation repose sur le travail de Srinivasan *et al.* [161] et revu par Kollman *et al.* [92]. Elle utilise le modèle de solvatation de Hawkins [64] avec les paramètres décrits par Tsui *et al.* [169].

Le calcul de l'énergie libre de liaison est basé sur le cycle thermodynamique suivant :

$$\begin{array}{lccc} \text{Phase gaz :} & \text{Protéine} + \text{Ligand} & \xrightarrow{\Delta G^{gaz}_{bind}} & \text{Protéine} \cdot \text{Ligand} \\ & \Big\downarrow \Delta G^{solv}_{prot} & \Big\downarrow \Delta G^{solv}_{lig} & \Big\downarrow \Delta G^{solv}_{comp} \\ \text{Phase aqueuse :} & \text{Protéine} + \text{Ligand} & \xrightarrow{\Delta G^{sol}_{bind}} & \text{Protéine} \cdot \text{Ligand} \end{array}$$

Ainsi, l'expression de ΔG^{sol}_{bind} est :

$$\Delta G^{sol}_{bind} = -\Delta G^{solv}_{prot} - \Delta G^{solv}_{lig} + \Delta G^{gaz}_{bind} + \Delta G^{solv}_{comp} \tag{3.22}$$

L'énergie libre de solvatation ΔG^{solv} est calculée en sommant une composante polaire ΔG^{solv}_{pol} et une composante non polaire ΔG^{solv}_{np}. La contribution non polaire est considérée comme étant proportionnelle à la surface accessible au solvant (SASA : *Solvant Accessible Surface Area*). Celle-ci est calculée à l'aide du logiciel Amber 8 grâce à l'algorithme "icosahedra". Deux méthodes peuvent-être utilisées pour calculer ΔG^{solv}_{pol} :

- par la résolution des équations de Poisson : *Molecular Mechanics Poisson Boltzman Surface Area* (MM–PBSA)
- par l'utilisation du modèle de Born : *Molecular Mechanics Generalized Born Surface Area* (MM–GBSA)

L'énergie libre de liaison hors solution, est donnée par la relation :

$$\Delta G^{gaz}_{bind} = E_{elec} + E_{vdw} + E_{int} - T(S_{comp} - (S_{prot} + S_{lig})) \tag{3.23}$$

avec E_{elec}, décrivant les forces électrostatiques, E_{vdw}, les forces de van der Waals et E_{int} l'énergie interne du système (terme géométrique).

Les termes entropiques sont décomposés en entropie translationnelle S_{trans}, rotationnelle S_{rot} et vibrationnelle S_{vib}. L'entropie translationnelle est dépendante de la masse des molécules, l'entropie rotationnelle dépend du moment d'inertie des molécules et l'entropie vibrationnelle est calculée par une analyse en modes normaux [165].

Flexibilité du récepteur

Le logiciel Dock 6.4 permet d'introduire de la flexibilité au sein du récepteur en accord avec la théorie de l'ajustement induit [93] (figure 3.6 page suivante). Le processus passe par une minimisation d'énergie, suivie par une dynamique moléculaire de Langevin à température constante, puis de nouveau une minimisation avant l'évaluation finale de l'énergie. Ces calculs utilisent le champ

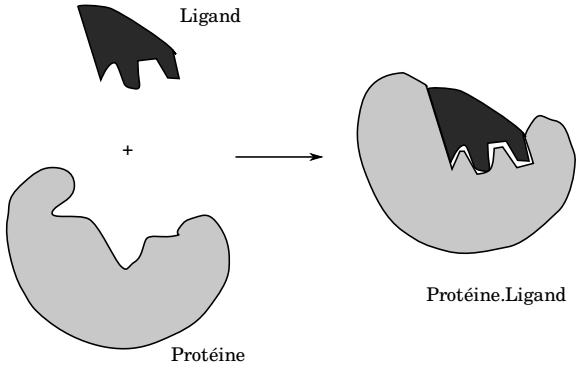


FIGURE 3.6 – Modèle de l'ajustement induit

de force AMBER [141, 176]. Le protocole utilisé est détaillé dans la publication de Graves *et al.* [57]. Les résidus à une distance inférieure ou égale à 3Å du ligand sont les seuls atomes flexibles.

3.2 Mise en place et évaluation d'un protocole de criblage virtuel

La modélisation permet l'obtention d'un modèle, c'est son but. Cependant le modèle est un reflet simplifié de la réalité donc il est intrinsèquement inexact. Nous avons vu que les algorithmes de *docking* sont des calculs empiriques donc simplifiés par rapport à la complexité physique du problème. Le caractère simplifiant du modèle est cependant bénéfique puisqu'il permet souvent de mieux comprendre le phénomène ou l'objet étudié, souvent complexe. Au fur et à mesure qu'un modèle se complexifie, du fait du progrès scientifique mais aussi technologique, il tend de manière asymptotique vers la réalité, à condition, bien entendu, qu'il soit construit en tenant compte de données expérimentales fiables. Ainsi le modèle vient de l'expérience, par exemple il nous faut, pour effectuer un *docking*, la structure tri-dimensionnelle du récepteur, et doit mener à l'expérience : les ligands sortis d'un criblage virtuel doivent être testés.

Enfin un modèle doit pouvoir être évalué. L'évaluation d'un modèle doit passer par la comparaison de celui-ci avec la réalité : les données expérimentales. Ainsi il est nécessaire pour évaluer un modèle d'avoir accès à un nombre important de données expérimentales afin que l'évaluation soit statistiquement satisfaisante. Dans notre cas, cette évaluation passe par l'utilisation de chimiothèques défiant les algorithmes de *docking* afin de se trouver

dans les conditions les plus difficiles à surmonter – en quelques sortes un banc d'essai pour le *docking*. Cette évaluation doit aussi être quantifiée et cette quantification doit pouvoir être comparée avec l'évaluation d'autres modèles.

3.2.1 Choix de la chimiothèque

La méthode du *docking* moléculaire est largement utilisée dans le domaine du criblage virtuel, cependant les algorithmes de *docking* ne sont pas parfait [135, 91]. Il est donc nécessaire d'évaluer de manière fiable ces algorithmes. Cette évaluation passe le plus souvent par le calcul de l'enrichissement : est évalué la capacité d'un logiciel de *docking* à classer en premier les molécules les plus actives. L'évaluation du protocole de *docking* nécessite donc l'utilisation d'une base de donnée moléculaire contenant deux sortes de molécules : des molécules d'activités connues contre la cible considérée et des molécules non-actives contre cette même cible. La capacité du logiciel de *docking* à retrouver les molécules actives est la propriété la plus souvent testée, l'aspect quantitatif de prévision de l'affinité est souvent ignoré. Le criblage virtuel par *docking* est donc un test binaire permettant de classer les molécules en deux groupes : actives et non-actives envers la cible considérée.

Le choix des molécules actives et non-actives ne doit pas être fait au hasard. En effet la fiabilité du test dépend grandement de la base de données choisie. Par exemple la fonction de score dépend souvent de la masse moléculaire des molécules [138]. De manière générale l'enrichissement observé peut être purement artificiel si les ligands présentent des propriétés physico-chimiques très différentes des molécules inactives [172]. Afin que l'enrichissement calculé soit réellement dû à des propriétés topologiques des ligands et non pas à des propriétés physiques triviales il est nécessaire que les molécules inactives présentent des propriétés physiques semblables aux ligands. On parle ainsi de leurre (*decoys* en anglais).

Les procédés de criblages virtuels présentés ici ont été réalisés sur différentes banques de molécules. Le logiciel AuPosSOM (voir chapitre 4 page 67 et publication [14]) a été initialement testé avec le logiciel AutoDock 4.0 sur trois cibles différentes : la protéase du VIH-1, la transcriptase inverse du VIH-1 et la thrombine humaine. Les coordonnées cartésiennes de ces récepteurs ont été téléchargées sur les serveurs de la *protein databank* (PDB). Pour chacune des cibles des molécules connues ont été extraites de la base de données : *binding database* (http ://www.bindingdb.org) [99] et associées à la base de données interne du laboratoire représentant ainsi une sous base de la chimiothèque nationale de France (http ://chimiotheque-nationale.enscm.fr), composée de 1 108 molécules. Pour les trois cibles, protéase, transcriptase inverse et thrombine, respectivement 22, 14 et 20 ligands connus ont été utilisés. Pour la trans-

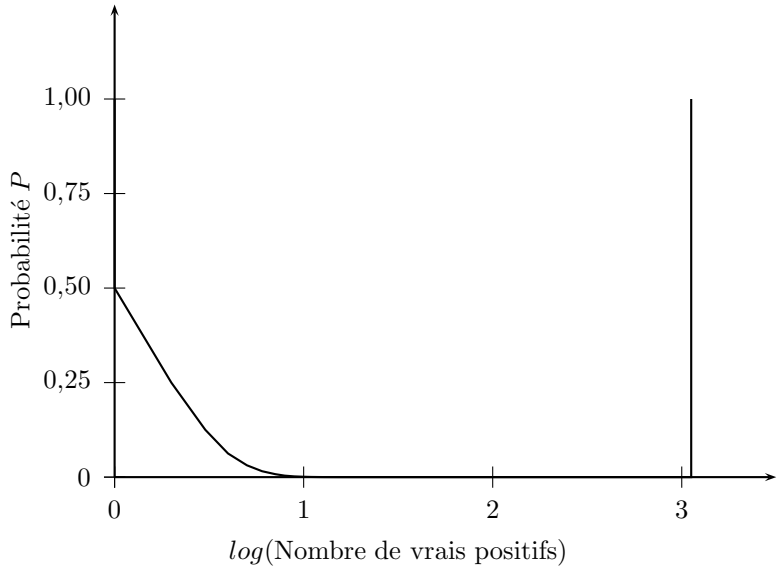


FIGURE 3.7 – Effet du nombre de vrais positifs dans une chimiothèque de 1 130 composés sur la probabilité de regrouper au moins l'ensemble des composés actifs. (voir équation 3.24)

criptase inverse uniquement des inhibiteurs non nucléosidiques (NNRTI) ont été utilisées.

Afin d'étudier l'influence du nombre de molécules actives introduites dans une banque de données lors des tests de protocole de criblage virtuel, la probabilité de sélectionner au moins l'ensemble des composés actifs par un processus aléatoire a été calculée. Cette probabilité P est donnée par la relation :

$$P = \frac{\sum_{i=0}^{n-p} C_{n-p}^{i}}{\sum_{j=p}^{n} C_{n}^{j}} \tag{3.24}$$

avec p le nombre de vrais positifs, n le nombre total de composés. L'évolution de cette probabilité en fonction du nombre de vrais positifs est donnée en figure 3.7. Pour le cas du criblage sur la protéase du VIH-1 $p = 22$ et $n = 1\,130$. Ainsi la probabilité de regrouper l'ensemble des composés actifs si nous considérons un processus aléatoire est de $2,38 \cdot 10^{-7}$. Toujours pour une banque de donnée de 1 130 composés la probabilité P est inférieure à 1% lorsque le nombre de vrais positifs est supérieur à six. Cette même probabilité devient inférieure à 1‰quand le nombre de vrais positifs est supérieur à dix. Ainsi en terme de rapport entre le nombre de vrais positifs et le nombre total de composés de la chimiothèque, le test réalisé est statistiquement pertinent.

Cependant pour les raisons expliqués ci-dessus et du fait que d'éventuelles

molécules actives puissent être présentes dans la base de données extraite de la chimiothèque nationale de France, il a été nécessaire de pousser plus loin le test des protocoles de criblage développés ici.

Pour ce faire nous avons utilisé la base de données DUD (*a Directory of Useful Decoys*) [73] afin d'avoir un test fiable des protocoles de criblage. Cette base de données accessible à l'adresse http ://dud.docking.org est composée de 2 950 composés actifs pour un total de 40 protéines cibles. Pour chaque composé actif, 36 leurres (decoys) – possédant des propriétés physiques similaires (masse moléculaire, logP), des topologies différentes et aucune activité ou affinité pour la cible considérée – sont présents. La base de données DUD est construite à partir de la base de données ZINC [75], une base de données libre de composés accessibles commercialement, créée pour le criblage virtuel.

Les distributions de cinq propriétés physiques (masse moléculaire, nombre d'accepteurs de liaison hydrogène, nombre de donneurs de liaison hydrogène, nombre de liaisons de libre rotation, logP) des ligands et des leurres de la DUD ont été tracées sur la figure 3.8 page ci-contre. Les propriétés des leurres sont comparables aux propriétés des ligands actifs à partir desquels ils ont été générés : la gamme de variation ainsi que la position du maximum est globalement la même.

3.2.2 Évaluation basée sur l'utilisation des courbes de ROC

L'acronyme ROC signifie *Receiver Operating Characteristic*. Cette représentation graphique issue des travaux théoriques de Neyman et Pearson [132, 133] a trouvé sa première application pratique lors de la seconde guerre mondiale. À cette époque les opérateurs radio britanniques utilisaient cette méthode afin de différencier les signaux correspondant à l'approche d'avions de guerre visant Londres par rapport à des signaux aléatoires correspondant à des interférences. Depuis lors, cette méthode est utilisée dans une multitude de domaines aussi variés que la médecine [183], l'acoustique [166], la météorologie [86], la criminologie [6], ... afin de déterminer la performance d'un modèle de classification binaire et/ou d'optimiser la classification par le modèle étudié.

Cependant l'utilisation des courbes de ROC n'est que récente dans le domaine du criblage virtuel et n'est pas nécessairement très répandue. Malgré tout son utilisation permet d'avoir une méthode standard d'évaluation et d'optimisation d'un outil de criblage *in-silico* [168].

Le but d'un criblage virtuel est de sélectionner par diverses méthodes calculatoires les molécules actives et aussi de dé-sélectionner les molécules

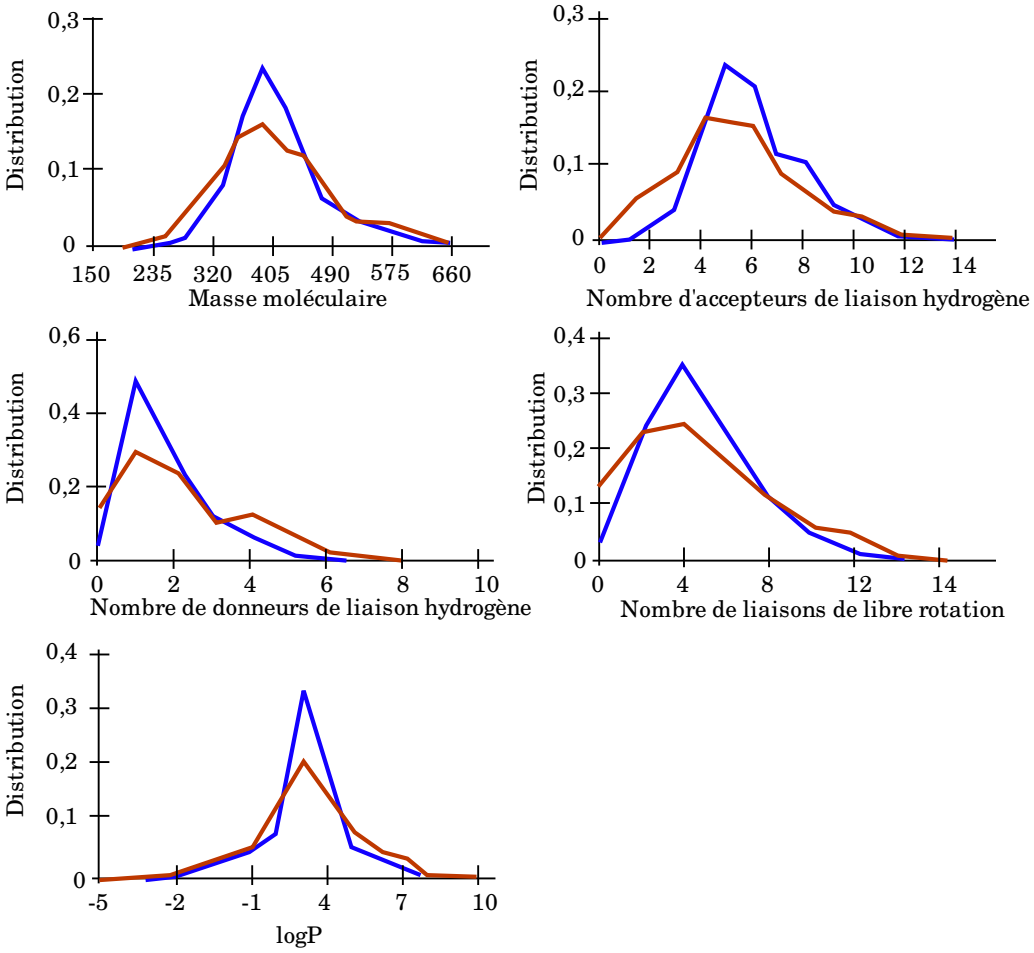


FIGURE 3.8 – Distribution des propriétés physiques des ligands (en bleu) et des leurres (en brun) de la DUD [73].

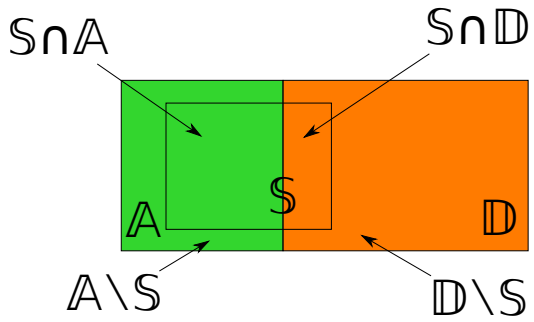


FIGURE 3.9 – Sous ensembles d'une chimiothèque à considérer lors d'un processus de criblage virtuel. L'ensemble $\mathbb{A}$ représente l'ensemble des molécules actives de la chimiothèque. L'ensemble $\mathbb{D}$ représente l'ensemble des molécules inactives de la chimiothèque. Ainsi $\mathbb{A} \cup \mathbb{D}$ représente l'ensemble de la chimiothèque. $\mathbb{S}$ représente l'ensemble sélectionné par le processus de criblage virtuel, ainsi $\mathbb{S} \cap \mathbb{A}$ est l'ensemble des vrais positifs, $\mathbb{S} \cap \mathbb{D}$ est l'ensemble des faux positifs, $\mathbb{A} \setminus \mathbb{S}$ est l'ensemble des faux négatifs et $\mathbb{D} \setminus \mathbb{S}$ est l'ensemble des vrais négatifs.

inactives d'une chimiothèque. Ainsi deux ensembles doivent-être formés : un ensemble sélectionné comportant le maximum de molécules actives (vrais positifs) et le minimum de molécules inactives (faux positifs) et son ensemble complémentaire correspondant devant comporter par conséquent le maximum de molécules inactives (vrais négatifs) et le minimum de molécules actives (faux négatifs) (voir figure 3.9).

Les ensembles définis en figure 3.9 permettent alors de calculer deux valeurs , la sensibilité (Se) et la spécificité (Sp), qui vont permettre de quantifier la qualité d'un criblage. La sensibilité (Se) est la proportion vrais positifs résultant du criblage, ainsi :

$$Se = \frac{card(\mathbb{S} \cap \mathbb{A})}{card(\mathbb{A})} \tag{3.25}$$

La spécificité (Sp) est la proportion de vrai négatif rejetée par le criblage, ainsi :

$$Sp = \frac{card(\mathbb{D} \setminus \mathbb{S})}{card(\mathbb{D})} = 1 - \frac{card(\mathbb{S} \cap \mathbb{D})}{card(\mathbb{D})} \tag{3.26}$$

Le nombre d'éléments et la composition de l'ensemble $\mathbb{S}$ sélectionné va dépendre d'une valeur seuil noté T (*threshold*). Dans des protocoles classiques de criblage virtuel, à chaque molécule peut être associé un score qui reflète l'énergie d'interaction avec le récepteur. Ainsi si on considère une valeur

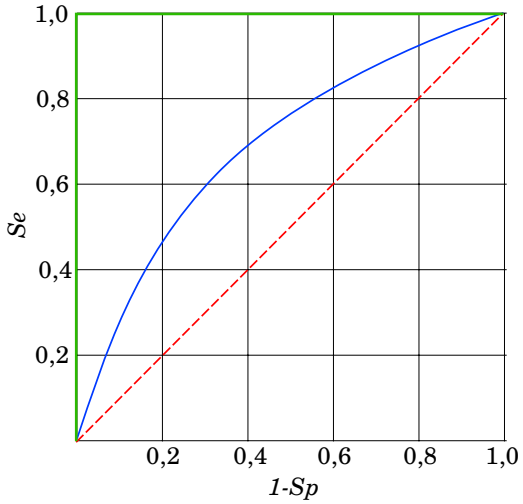


FIGURE 3.10 – Espace de ROC. Le tracé rouge représente le résultat pour un processus de sélection aléatoire. Le tracé bleu représente une courbe de ROC d'un processus de sélection discriminant en partie les molécules actives au sein d'une base de donnée. Le tracé vert représente le cas d'un processus de sélection idéal.

d'énergie d'interaction nous cherchons à sélectionner les molécules qui ont une énergie la plus basse possible (l'énergie libre de formation du complexe devant être négative), correspondant à la structure la plus stable. Pour un seuil T donné nous cherchons ainsi les molécules dont l'énergie libre ΔG est inférieure à T. Si T est supérieure à l'ensemble des énergies calculées alors l'ensemble des molécules est sélectionné, dans ce cas $\mathbb{S} = \mathbb{A} \cup \mathbb{D}$, c'est à dire que l'ensemble des molécules est sélectionné (seuil trop permissif) donc $Se = 1$ et $Sp = 0$. Au contraire si T est inférieur à l'ensemble des énergies alors aucune molécule n'est sélectionné, dans ce cas $\mathbb{S} = \emptyset$ (seuil trop restrictif) donc $Se = 0$ et $Sp = 1$. Ainsi pour une cible donné nous devons trouver le seuil optimum (T_{opt}) tel que Se et Sp soient maximum. Pour ce faire, il est possible de tracer l'évolution du taux de vrai positif (TPR : *True Positive Rate*) c'est à dire la sensibilité Se en fonction du taux de faux positifs (FPR : *False Positive Rate*) soit $1 - Sp$. Le tracé $Se = f(1 - Sp)$ représente une courbe de ROC (voir figure 3.10). L'espace de ROC est définit dans l'intervalle de définition $[0; 1]^2$.

Afin de quantifier la qualité d'un processus de criblage virtuel il est intéressant de calculer l'aire sous la courbe de ROC ou AUC : *Area Under Curve*. En effet cette valeur donne la probabilité qu'un composé actif choisi

aléatoirement soit classé au dessus d'un composé inactif choisi aléatoirement [45]. Ainsi pour un processus dont la courbe de ROC est confondue avec la diagonale (en rouge sur la figure 3.10 page précédente), l'AUC est égale $0,5$ ce qui correspond donc à un processus aléatoire. Ainsi un modèle capable de différencier des composés actifs parmi des composés inactifs est représenté par une courbe qui passe au dessus de la diagonale (en bleu sur la figure 3.10 page précédente), dont l'AUC est par conséquent supérieure à $0,5$. Un modèle idéal, fiable à 100%, est caractérisé par une AUC de 1 et possède donc une courbe qui se confond avec les intervalles de définition y et x de l'espace de ROC (tracé vert sur la figure 3.10 page précédente).

Le choix du meilleur seuil (T) doit être effectué en sélectionnant le point le plus proche du point idéal de coordonnées $(0,1)$. Afin de choisir ce point nous proposons de calculer la norme du vecteur $\vec{\gamma}$ qui relie le point idéal $(0,1)$ à un point de la courbe de ROC. Cette norme est donnée par la relation :

$$\|\vec{\gamma}\| = \sqrt{(1-Sp)^2 + (1-Se)^2} \tag{3.27}$$

Le point associé au $\|\vec{\gamma}\|$ le plus faible correspond au point associé au seuil optimal T.

L'algorithme 1 page 64 permet de calculer les coordonnées des points de la courbe de ROC correspondant à un résultat de criblage virtuel. Ce résultat consiste en une série de réels, correspondant à des valeurs de scores, associés à l'identifiant d'une molécule. Afin de générer de manière efficace les points de la courbe de ROC à partir de cette liste, nous pouvons utiliser le caractère monotone de la classification binaire par le score. En effet, si une molécule est active pour un seuil donné, cette molécule restera active pour un seuil inférieur. Si une molécule est inactive, elle restera inactive quelque-soit le seuil considéré. Ainsi en classant les molécules par scores décroissants il est aisé et rapide de calculer les points de la courbe de ROC en parcourant un à un les éléments de la liste ainsi triée. Si nous commençons par une molécule active il suffit de compter une à une ces molécules en parcourant la liste jusqu'à l'obtention d'une molécule inactive. Alors dans ce cas nous obtenons le premier point de la courbe de ROC d'abscisse 0 et d'ordonnée y_1. Ensuite nous comptons pas à pas, en partant de cette molécule inactive, le nombre de molécules inactives jusqu'à l'obtention d'une molécule active. Nous obtenons alors un deuxième point d'abscisse x_1 et d'ordonnée y_1. Nous pouvons alors reprendre le dénombrement des molécules actives jusqu'à ce que nous rencontrions une molécule inactive : nous obtenons alors le point de coordonnée (x_1, y_2). En itérant cette procédure nous traçons ainsi la totalité de la courbe de ROC. Si nous partons avec une molécule inactive la procédure est la même sauf que le premier point est alors d'ordonnée nulle et d'abscisse

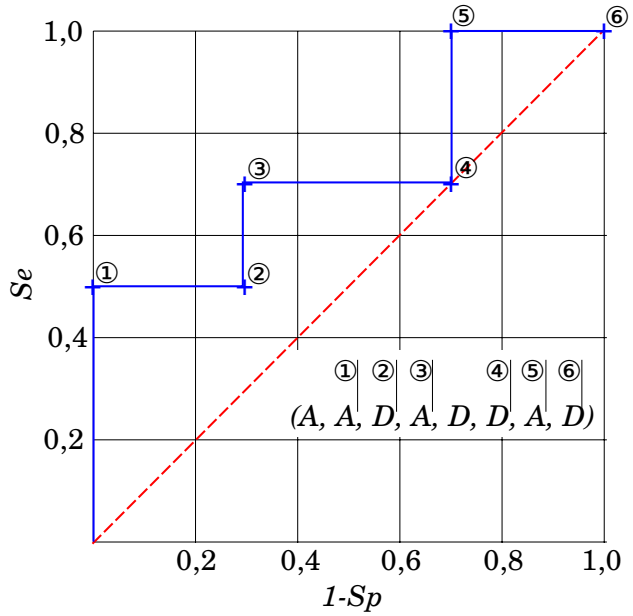


FIGURE 3.11 – Illustration de l'algorithme de construction d'une courbe de ROC (voir algorithme 1 page suivante). Le vecteur (A, A, D, A, D, D, A, D) représente une liste triée selon le score, de molécules actives (A) et inactives (D). Les points numérotés de 1 à 6 indiquent les étapes du tracé de la courbe de ROC en accord avec l'algorithme 1 page suivante.

non nulle (x_1). Donc à chaque changement du caractère actif ou non actif de la molécule il faut enregistrer, au sein d'un vecteur, les coordonnées d'un point de ROC. Cette procédure est présentée dans l'algorithme 1 page suivante et illustrée par la figure 3.11.

Algorithme 1 (Méthode de génération des points de ROC)
Nous considérons ici une fonction de score arbitraire qui attribue aux complexes les plus stables un score positif maximal.
Entrées : *M, liste des molécules étudiées ; $f(M)$, est une fonction permettant de déterminer si la molécule M est active $(f(i) = 1)$ ou inactive $(f(i) = 0)$; $P > 0$ le nombre de molécules actives ; $N > 0$ le nombre de molécules inactives.*
Sorties : *R, la liste des coordonnées des points de la courbe de ROC.*

$M_{trié} \xleftarrow{\text{Tri par ordre décroissant de score}} M$
$FP \leftarrow TP \leftarrow 0$ *(Nombre de faux positifs : FP ; et de vrais positifs : TP)*
$R \leftarrow [\,]$

$f_{prec} \leftarrow -1$
$i \leftarrow 1$
tant que $i \leq |M_{trié}|$ ***:***
    si $f(i) \neq f_{prec}$ ***:***
        Ajouter les valeurs $(\frac{FP}{N}, \frac{TP}{P})$ *à* R
        $f_{prec} \leftarrow f(i)$
    si $f(i) = 1$ ***:*** *#(c'est à dire que la molécule considérée est active)*
        $TP \leftarrow TP + 1$
    sinon :
        $FP \leftarrow FP + 1$
    $i \leftarrow i + 1$
Ajouter les valeurs $(\frac{FP}{N}, \frac{TP}{P})$ *à* R *#(ajout des derniers points)*

3.2.3 Réalisation

Équipements utilisés

Afin de réaliser les calculs de *docking* dans une période de temps raisonnable les calculs ont été réalisés pour la plupart sur le *cluster* de calcul de la plateforme bioinformatique MIGALE de l'unité "Mathématique, Informatique et Génomique" INRA [5]. Le cluster de la plateforme est constitué de 376 équivalents processeurs de génération différentes, allant du Xeon 64 bits au Quad Core (128 coeurs AMD Quad Core 8354 2.8 GHz ; 128 coeurs Intel Quad Core E5520 2.27 GHz ; 80 coeurs Intel Quad Core 5340 2.33 GHz ; 40 coeurs Intel Dual Core 5140 2.33 GHz). La couche logicielle permettant d'exploiter au mieux cette puissance de calcul est un gestionnaire de file d'attente nommé Sun Grid Engine. Le *cluster* de la plateforme est de type Beowulf.

Pour des calculs moins importants 10 pseudo-processeurs ont été mis en parallèle au sein du laboratoire (Intel Core2 Duo CPU P8400 @ 2.26GHz ; Intel Core2 Quad CPU Q6700 @ 2.66GHz ; Intel Core2 Quad CPU Q9400 @ 2.66GHz) en utilisant la couche logicielle *parallel python* :
(http ://www.parallelpython.com).

Préparation des molécules

Les charges partielles atomiques de la protéine ont été calculées grâce au champs de force AMBER ff99SB [30] pour les résidus d'acides aminés standards. Les charges partielles atomiques des molécules dockées ont été calculées avec le module Antechamber du programme AMBER avec la méthode AM1-BCC [175, 176].

5. Institut National de Recherche Agronomique

Protéine	Code PDB	Référence bibliographique
Protéase du VIH-1	2bpw	[131]
Transcriptase inverse du VIH-1	1uwb	[33]
	1rt4	[42]
	2be2	[68]
	1jla	[148]
	1rt1	[72]
Thrombine humaine	1afe	[36]
Complexe IN·LEDGF	2b4j	[23]
Domaine catalytique de l'intégrase	1bl3	[109]
Domaine catalytique de l'intégrase du virus du sarcome aviaire	1vsh	[15]

TABLE 3.1 – Code d'accès aux principales structures tridimensionnelles étudiées.

Les molécules dockées ont préalablement étaient minimisées avec le programme MMTK (*Molecular Modelling Toolkit*) [69] avec 10 000 étapes de minimisation et un pas de 0,02Å. Le module Antechamber [174] de AMBER a été utilisé pour attribuer les paramètres atomiques à la molécule.

Les codes d'accès aux fichiers de coordonnées atomiques sur la *protein databank* (http ://www.pdb.org) sont donnés dans le tableau 3.1.

Chapitre 4

Développement et implémentation du logiciel AuPosSOM : Automatic Analysis of Poses using Self-Organizing Map

If the brain were so simple that we could understand it then we'd be so simple that we couldn't.

Lyall Watson

4.1 Introduction

Le criblage virtuel consiste en une série de filtre de diverses natures ayant pour but de diminuer progressivement la liste des composés potentiellement actifs. La diminution de cette liste doit être faite de telle sorte que le taux de vrais positifs augmente, en même temps que le taux de faux positifs diminue. Le processus de criblage commence en général par une évaluation rapide des composés vis à vis de leur toxicité et de leur potentialité à être un composé actif (filtre ADME-Tox : *Absorption Distribution Metabolism Elimination Toxicity*). Le processus continue ensuite par un criblage basé sur la structure des ligands (*ligand-based approaches*) et/ou basé sur la structure du récepteur (*Structure-based approaches*) [129].

Pour l'approche basée sur la structure, les différentes molécules de la base de données sont dockées sur la structure tridimensionnelle du récepteur.

L'amélioration des protocoles de *docking* (voir section 3.1 page 41) permet la génération de conformations similaires aux données cristallographiques. Cependant les fonctions de score n'arrivent pas toujours à discriminer la pose correspondant à la pose cristallographique vis à vis de l'ensemble des poses générées par l'algorithme de *docking* [179]. L'amélioration des fonctions de score reste un enjeux important dans le domaine du criblage virtuel basé sur le *docking*. Cependant, un filtre de score permet de réduire sensiblement le nombre de molécules candidates, mais le classement de ces molécules ne peut être effectué par l'utilisation du score. L'inspection visuel des poses peut permettre de restreindre cette liste, mais celle ci est impossible à réaliser lorsque le nombre de molécule est supérieur à la centaine [87, 4]. De plus cette étape est dépendante des connaissances de l'utilisateur dans le domaine et reste donc très subjective et fastidieuse. Ainsi d'autres filtres doivent être appliqués afin de réduire la liste de molécules.

Le logiciel AuPosSOM [1] propose une méthode de regroupement de molécules grâce à l'utilisation de vecteurs mathématiques décrivant les interactions entre ligand et récepteur. Les données sont regroupés par une carte auto-organisatrice de Kohonen ou SOM (*Self-Organizing Map*) [90]. Ce réseau de neurones artificiels permet de réduire l'espace dimensionnel. Les vecteurs à n dimensions sont cartographiés sur un espace à deux voire trois dimensions. Ces structures intelligentes de représentation de données sont inspirées par les neurosciences, comme la plupart des algorithmes développés en intelligence artificielle. Il s'agit ici de reproduire le principe neuronal du fonctionnement du cerveau des vertébrés : des stimuli de même nature excitent une même région corticale (voir homonculus de Penfield [156], figure 4.1 page ci-contre). De la même manière, la carte auto organisatrice se déploie de façon à représenter un ensemble des données, et chaque neurone se spécialise pour représenter un groupe bien particulier des données selon les points communs qui les rassemblent (figure 4.2 page suivante). Renner *et al.* [149] a développé une méthode de *scoring* consensus utilisant des empreintes de contacts et une analyse par carte de Kohonen. Cependant il n'est pas proposé un protocole d'analyse de la carte de Kohonen permettant de classé hiérarchiquement les molécules de façon non supervisée.

Le classement hiérarchique des molécules suivant les poses de *docking* propose une alternative au problème des fonctions de score et de l'identification de modes alternatifs de liaisons des molécules.

De plus, l'analyse proposée ici repose sur une analyse statistique de l'ensemble des poses générées par les logiciels de *docking*. Ainsi le problème de la sélection de la meilleure pose est contourné puisque l'ensemble des poses

1. *Automatic analysis of Poses using Self-Organizing Map*

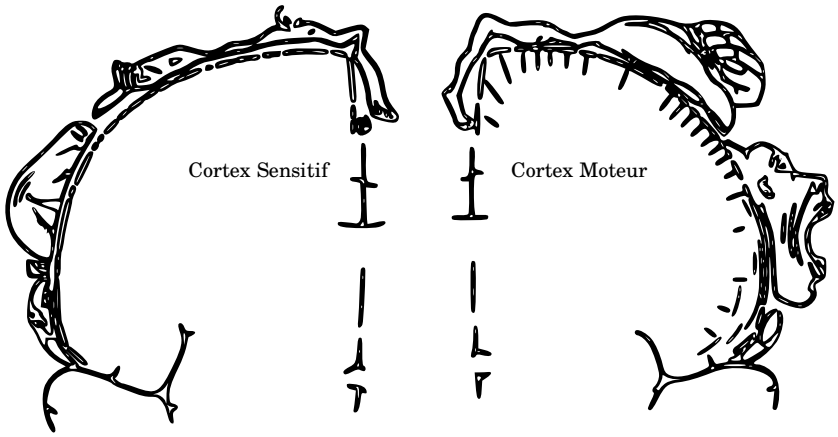


FIGURE 4.1 – Homonculus de Penfield : la partie de gauche représente la topographie du cortex sensitif et la partie de droite du cortex moteur. Sur la surface du cortex est représentée la partie du corps qui est activée si on stimule la zone correspondante. Ainsi, un espace à trois dimensions – le corps humain – est représenté sur une surface bidimensionnelle – la surface corticale. Cette somatotopie correspond à une diminution de l'espace dimensionnel comme pour les cartes de Kohonen.

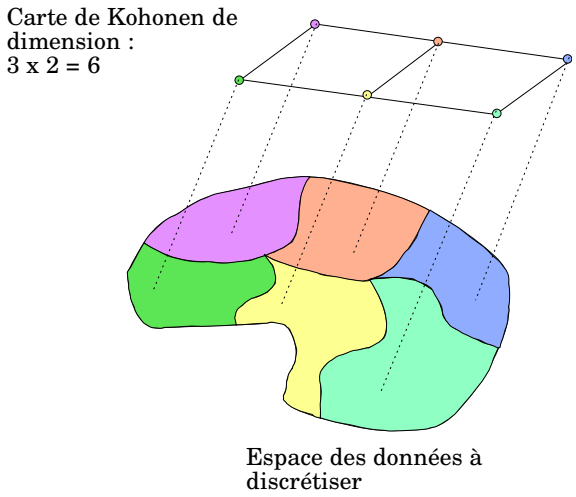


FIGURE 4.2 – La carte de Kohonen permet de représenter un espace complexe à n dimensions dans un espace de dimension réduite p, avec $p < n$ (ici $p = 6$). L'espace ainsi discrétisé est découpé en p ensembles chaque ensemble contenant x données présentant des points communs qui les rassemble. Chacun de ces ensembles est représenté par un point dans la carte de Kohonen.

générées est pris en compte. Le processus itératif lié à l'entraînement des cartes de Kohonen permet de supprimer les poses isolées correspondant à des modes de liaison non significatifs (l'ensemble de ces poses est nommé "bruit" par analogie avec les techniques de traitement du signal). L'analyse des cartes de Kohonen est souvent délicate, notamment afin de déterminer les frontières entres groupes. Une méthode récente d'analyse non supervisée des cartes de Kohonen, développée par Samsonova *et al.* [153] a été implémentée au sein du logiciel AuPosSOM.

Ainsi les molécules sont classées suivant leur mode de liaison au récepteur. Si des composés actifs sont introduits dans la chimiothèque criblée contenant des composés d'activités inconnues, les composés actifs sont regroupés avec un nombre restreint de composés de la chimiothèque, ainsi potentiellement actifs. La notion de CAR (*Contact Activity Relationship*) est ainsi introduite et validée.

L'architecture globale de l'algorithme développé peut être représentée par le diagramme en figure 4.3 page ci-contre. L'ensemble du programme AuPosSOM a été implémenté en langage Python [152].

4.2 Approche théorique

Afin d'appréhender le plus clairement possible cette partie nous allons nous baser sur le diagramme présenté en figure 4.3 page suivante et détailler chaque étape du processus du point de vue théorique et pratique.

4.2.1 Génération des vecteurs

La génération des vecteurs est une étape importante dans le processus global de classement des molécules. En effet ce sont ces vecteurs qui décrivent individuellement chaque molécule. Pour chaque molécule les n poses sont analysées en terme de liaisons hydrogènes et de contacts hydrophobes. Les vecteurs ainsi obtenus sont moyennés sur l'ensemble des n poses. Nous obtenons ainsi deux vecteurs : un vecteur décrivant les liaisons hydrogènes et un vecteur décrivant les contacts hydrophobes. Ces vecteurs sont ensuite entrelacés : le premier élément du vecteur décrit les liaisons hydrogènes avec le premier acide aminé contacté, le deuxième élément décrit les contacts hydrophobes avec ce même acide aminé – le troisième élément décrit les liaisons hydrogènes avec le deuxième acide aminé contacté, le quatrième élément les contacts hydrophobes et ainsi de suite, comme le montre le diagramme sui-

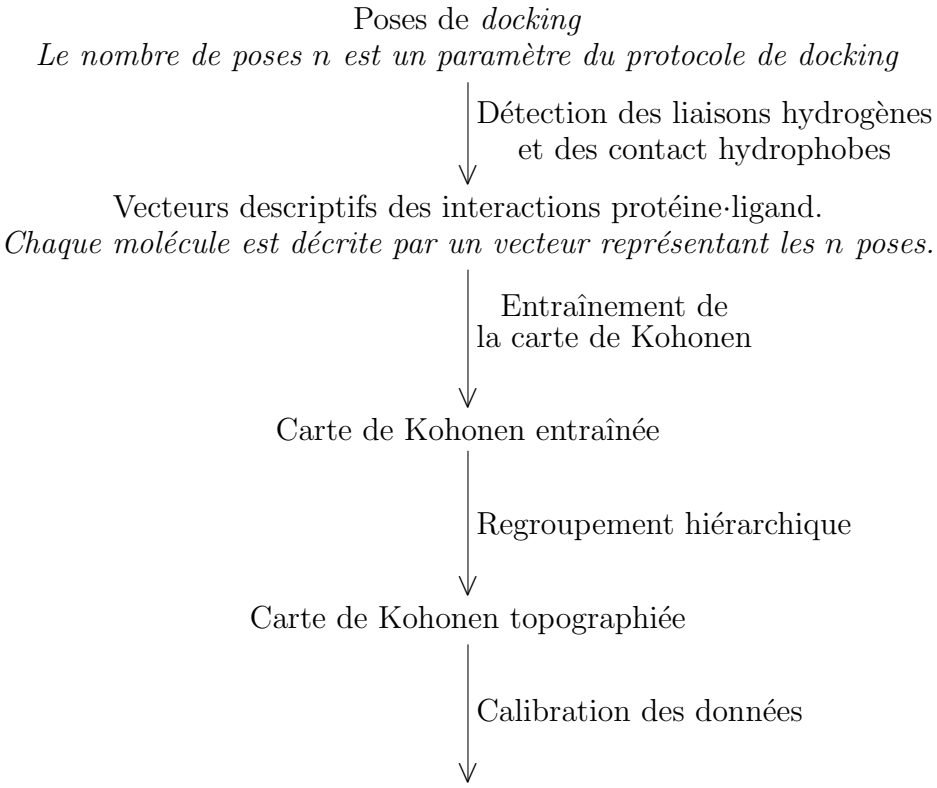
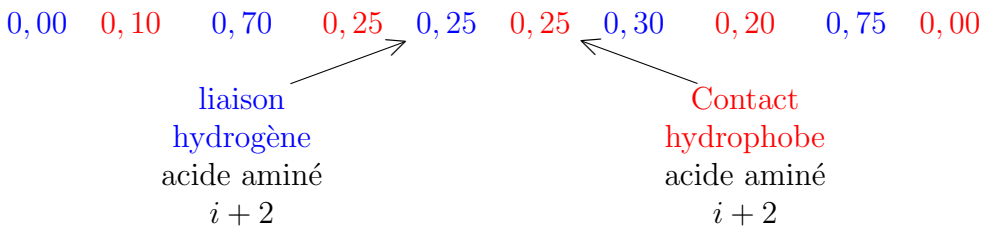


FIGURE 4.3 – Architecture globale du logiciel AuPosSOM

vant :



Dans cet exemple, 0,25 indique que l'acide aminé $i + 2$ effectue une liaison hydrogène dans 25% des poses de *docking*, de même pour les contacts hydrophobes. Afin de diminuer le temps de calcul, et d'économiser de la mémoire vive lors de l'entraînement de la carte de Kohonen, les vecteurs sont simplifiés. Ce temps est diminué considérablement du fait de la diminution importante de la dimension des vecteurs, ainsi les éléments matriciels de la carte de Kohonen sont aussi de dimension moindre. Après génération de l'ensemble des vecteurs les éléments vectoriels étant toujours égaux à zéro sont supprimés. Le code source python permettant la génération des vecteurs est présenté en page 195.

Initialement, la détection des liaisons hydrogènes et des contacts hydrophobes était réalisée par le logiciel LigPlot [173] qui utilise lui même HBPlus [117] pour le calcul des liaisons hydrogènes. Un script bash permettait d'automatiser la génération des vecteurs à partir des fichiers au format pdb comportant à la fois les coordonnées atomiques du ligand et de la protéine (format d'entrée du logiciel LigPlot). Plusieurs contraintes sont apparues suite au développement de ce processus : l'utilisation du format pdb, peu flexible et la difficulté de définir la topologie du ligand dans ce format, et la dépendance au logiciel LigPlot qui rend la modification et l'adaptabilité du programme difficile. De plus les logiciels de *docking* utilise souvent le format mol2 pour le ligand et le format pdb pour la protéine. Le résultat du *docking* consiste ainsi en une série de fichiers au format mol2 ou en un unique fichier mol2 contenant les coordonnées atomiques de l'ensemble des ligands dans le repère de la protéine. Ainsi la génération des formats d'entrées pour LigPlot était laborieuse et source d'erreur : conversion du mol2 en pdb et concaténation du pdb du ligand et de la protéine. Par conséquent l'information de structure de la protéine était redondante ce qui augmentait considérablement l'utilisation de l'espace disque.

Afin de s'affranchir de la dépendance au logiciel LigPlot, un algorithme de calcul des liaisons hydrogènes et des contacts hydrophobes a été développé [14] (voir code source page 202 et 209). Ce script initial, écrit en python utilise le module Bio.PDB [58] du projet Biopython (http ://biopython.org). La liste des donneurs et des accepteurs de liaisons hydrogène est générée

pour le ligand et pour la protéine. Une interaction est considérée comme une liaison hydrogène lorsque la distance D−H· · ·A est de $1,85 \pm 0,65$Å et l'angle D−H· · ·A de $180 \pm 80°$ et une interaction est considérée comme hydrophobe lorsque deux atomes hydrophobes sont distant de moins de $3,9$Å. Afin d'évaluer la rapidité de cet algorithme le code a été testé sur 20 000 structures (1000 molécules avec 20 poses par molécule). Sur un PC Linux (quadricore Intel 2.66GHz, 1GB RAM) ce calcul prend 58 minutes soit une moyenne de 0,174 seconde par structure. Cependant, du fait de l'utilisation du module Bio.PDB, cette méthode est toujours dépendante de fichiers pdb. Pour les mêmes raisons que celle développées ci dessus, un autre protocole de génération des vecteurs a été proposé.

Ce protocole est basé sur le logiciel UCSF-Chimera [142]. Ce logiciel de visualisation écrit en python contient un grand nombre de fonctions permettant de calculer diverses propriétés physico-chimiques et en particulier les liaisons hydrogènes et les contacts hydrophobes, en lisant à la fois les fichiers pdb ou mol2. Un langage de script permet d'interfacer ce logiciel avec des programmes écrit en langage python et est donc bien adapté au problème posé. Ainsi le code source présenté en page 211 permet de réaliser les calculs de liaison hydrogènes et contacts hydrophobes. Par défaut deux fichiers d'entrée sont fournis : un fichier mol2 contenant les coordonnées atomiques du récepteur et un fichier mol2 contenant les coordonnées des différents ligands dockés avec l'ensemble des poses ("multimol2"). Le script permettant la création et la simplification des vecteurs ainsi que le formatage du fichier de vecteurs est présenté en page 212. Avec ce protocole l'élément au sein d'un vecteur ne représente pas un acide aminé mais un atome. Ainsi l'analyse est plus fine et représente le nombre de contact moyen par atome au sein de la protéine.

Le formatage du fichier de vecteur obtenu est identique à celui décrit par Samsonova *et al* [153]. À chaque ligne correspond un vecteur et à chaque fin de ligne est indiquée l'identifiant de la molécule considérée. La première ligne donne le nombre d'élément de chaque vecteur, chacun devant contenir le même nombre d'éléments. La figure suivante donne un exemple de fichier de vecteurs généré par AuPosSOM :

scripts/ALL_vectors

```
1 250
2 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.03 0.03 0.00
      0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
    0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
    0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
    0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.03 0.03 0.24
    0.24 0.06 0.06 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.06
```

	0.06 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.09 0.09 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.03 0.03 0.00 0.00 0.00 0.00 0.12 0.12 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.03 0.03 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.03 0.03 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.18 0.18 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.09 0.09 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.06 0.06 0.09 0.09 0.00 0.00 0.03 0.03 0.00 0.00 ZINC00018097
3	0.01 0.01 0.01 0.01 0.01 0.01 0.00 0.00 0.01 0.01 0.03 0.03 0.03 0.03 0.04 0.04 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.05 0.05 0.00 0.00 0.00 0.00 0.02 0.02 0.05 0.05 0.01 0.01 0.00 0.00 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.10 0.10 0.00 0.00 0.01 0.01 0.03 0.03 0.00 0.00 0.04 0.04 0.04 0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.04 0.04 0.04 0.04 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.02 0.02 0.03 0.03 0.01 0.01 0.03 0.03 0.01 0.01 0.01 0.01 0.01 0.01 0.00 0.00 0.03 0.03 0.01 0.01 0.00 0.00 0.01 0.01 0.01 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.02 0.02 0.01 0.01 0.03 0.03 0.01 0.01 0.02 0.02 0.01 0.01 0.02 0.02 0.06 0.06 0.01 0.01 0.00 0.00 0.01 0.01 0.01 0.01 0.01 0.01 0.02 0.02 0.03 0.03 0.01 0.01 0.02 0.02 0.03 0.03 0.01 0.01 0.00 0.00 0.01 0.01 0.01 0.01 0.03 0.03 0.02 0.02 0.00 0.00 0.00 0.00 0.03 0.03 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.00 0.00 0.01 0.01 0.01 0.01 0.00 0.00 0.01 0.01 0.01 0.01 0.03 0.03 0.01 0.01 0.01 0.01 0.01 0.01 0.00 0.00 0.00 0.00 0.04 0.04 0.01 0.01 0.01 0.01 0.00 0.00 0.00 0.00 0.04 0.04 0.00 0.00 0.04 0.04 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.02 0.02 0.01 0.01 0.02 0.02 0.01 0.01 0.01 0.01 0.04 0.04 0.05 0.05 0.01 0.01 0.12 0.12 0.01 0.01 ZINC00538635

Par la suite chaque vecteur d'entrée sera noté V.

4.2.2 Carte de Kohonen

C'est en 1943 que McCulloch et Pitts ont présenté pour la première fois le concept de neurone formel abstraction du neurone physiologique [116]. En 1958 Rosenblatt développe le modèle du perceptron [151], réseau de neurone supervisé inspiré du système visuel. C'est le premier système capable d'apprendre par expérience. En 1982, Hopfield propose un réseau bouclé sur lui même, c'est le réseau bouclé d'Hopfield [71]. C'est en 1997 que Kohonen pro-

pose son système de carte auto-organisatrice ou *Self-Organizing Map* (SOM) [90], réseau neuronal non supervisé.

Par opposition aux réseaux de neurones supervisés, la sortie correcte ne peut pas être défini, *a priori*. La carte de Kohonen permet de cartographier l'espace des entrées souvent complexe (contenant un grand nombre d'éléments – vecteurs –) en un espace plus simple contenant un nombre restreint d'éléments. Chaque vecteur de l'espace d'entrée ne va "exciter" qu'un seul type de neurone (élément matriciel de la carte de Kohonen) (voir figure 4.2 page 69). Cette simplification de l'espace d'entrée permet de représenter une image plus simple de celui-ci en regroupant les données proches.

Architecture de la carte de Kohonen

La carte de Kohonen est une matrice de dimension $X \times Y$ avec X et Y le nombre de colonnes et de lignes respectivement :

$$M = \begin{pmatrix} \Omega_{11} & \Omega_{12} & \cdots & \cdots & \Omega_{1\pi} \\ \Omega_{21} & \Omega_{22} & \cdots & \cdots & \Omega_{2\pi} \\ \vdots & \ddots & & & \\ & & \Omega_{ij} & & \\ \vdots & & & \ddots & \\ \Omega_{(\nu-1)1} & \Omega_{(n-1)2} & \cdots & \cdots & \Omega_{(\nu-1)\pi} \\ \Omega_{\nu 1} & \Omega_{n2} & \cdots & \cdots & \Omega_{\nu\pi} \end{pmatrix} \tag{4.1}$$

avec $X = \pi$ et $Y = \nu$.

Chaque élément matriciel Ω_{ij} (neurone) est un vecteur de la même dimension que les vecteurs d'entrée V. Chaque élément du vecteur est noté ω_{ijk}, le couple (i, j) notant la position dans la matrice M et k la position au sein du vecteur Ω_{ij}.

$$\Omega_{ij} = (\omega_{ij1}, \omega_{ij2}, \ldots, \omega_{ijk}, \ldots, \omega_{ijn}) \tag{4.2}$$

avec $n = card(V)$. La matrice est initialisée aléatoirement avec des vecteurs (Ω_{ij}) dont les éléments sont compris entre l'élément minimum de l'ensemble des éléments des vecteurs d'entrées V et l'élément maximum de l'ensemble des éléments des vecteurs V. Le code suivant extrait de l'annexe page 217 montre l'implémentation python d'initialisation de la carte de Kohonen.

```
1 import RandomArray
2 # Kohonen map initialization
```

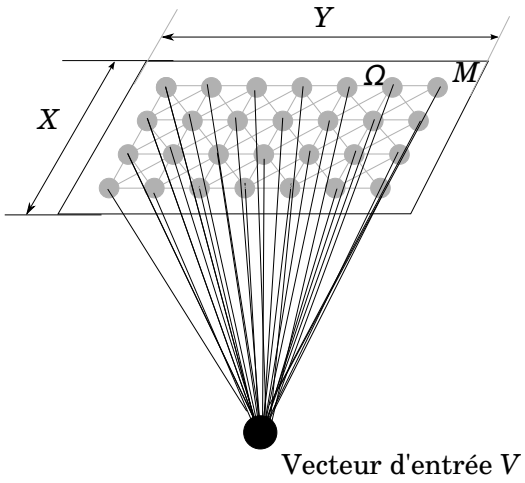


FIGURE 4.4 – Connections entre le vecteur d'entrée V et les éléments Ω de la carte de Kohonen M.

```
3 self.cardinal = len(self.inputvectors[0]) # card(V)
4 maxinpvalue = max([max(v) for v in self.inputvectors]) # Maximal
      element of all vector's elements
5 mininpvalue = min([min(v) for v in self.inputvectors]) #
      Minimal element of all vector's elements
6 self.M = RandomArray.uniform(mininpvalue,maxinpvalue,(self.X,
      self.Y, self.cardinal)) # Kohonen map initialization with
      dimensions (self.X, self.Y). Each element is a vector with a
      dimension equal to self.cardinal
```

Les vecteurs d'entrée agissent sur l'ensemble des éléments Ω_{ij} de la carte de Kohonen M, et les éléments Ω interagissent de même entre eux, comme le montre la figure 4.4. C'est l'ensemble de ces inter-connections qui permet l'entraînement de la carte de Kohonen.

4.2.3 Entraînement de la carte de Kohonen

Les paramètres permettant l'entraînement de la carte de Kohonen sont entrés par l'utilisateur dans le fichier `AuPosSOM.conf`. Le fichier créé par défaut par l'application est le suivant :

```
1 # Map parameters
2 X                    = 5
3 Y                    = 4
4
```

```
5  # Training
6  number_of_phase     = 2     # default 2 phases
7
8  # Training Phase 1
9  alpha_begin_1       = 0.2
10 alpha_end_1         = 0
11 radius_begin_1      = 6
12 radius_end_1        = 1
13 iterations_1        = 1000
14
15 # Training Phase 2
16 alpha_begin_2       = 0.02
17 alpha_end_2         = 0
18 radius_begin_2      = 3
19 radius_end_2        = 1
20 iterations_2        = 10000
```

Ce fichier regroupe les paramètres de la carte (dimensions X et Y), ainsi que les paramètres d'apprentissage ou entraînement, à savoir le taux d'apprentissage initial $\alpha(0)$ et final $\alpha(t_f)$, le rayon initial $\sigma(0)$ et final $\sigma(t_f)$ et le nombre d'itérations $t = \{0, 1, 2, \cdots, t_f\}$ pour les deux phases d'apprentissage. L'algorithme 2 décrit l'algorithme d'entraînement des cartes de Kohonen.

Algorithme 2 (Entraînement des Cartes de Kohonen)
Cet algorithme se décompose en quatre étapes :

Initialisation *Les éléments Ω_{ij} de dimensions n de la matrice M sont initialisés aléatoirement comme décrit en page 75. Initialisation du compteur d'itération t à la valeur $0 : t = 0$. Lecture du fichier de paramètres décrit page 76 et initialisation de $\alpha(0)$, $\alpha(t_f)$, $\sigma(0)$, $\sigma(t_f)$.*

Étape 1 *Sélection aléatoire d'un vecteur d'entrée V.*

Étape 2 *L'élément Ω de la matrice M le plus proche en terme de distance euclidienne est sélectionné.*

Étape 3 *Mis à jour des variables α et σ en accord avec t : $\alpha \leftarrow \alpha(t)$ et $\sigma \leftarrow \sigma(t)$. Les vecteurs de poids sont modifiés en accord avec la règle de modification des poids la fonction de voisinage $\Theta(t)$ et α.*

Étape 4 *Arrêt si le nombre d'itérations est atteint ($t = t_f$) ; sinon $t \leftarrow t + 1$ et retour à l'**Étape 1**.*

Nous allons détailler ci-dessous l'ensemble des étapes de l'algorithme d'entraînement des cartes de Kohonen.

Étape 1 : Sélection des vecteurs d'entrée

Le code ci dessous extrait du code source page 217 est l'implémentation python de l'algorithme 2 page précédente. La sélection aléatoire des vecteurs se fait par tirage successif et sans remise d'un index identifiant un vecteur d'entrée V.

```
1   def learn(self):
2     Map = self.M
3     kv = range(len(self.inputvectors)) # List of input vector
          identifiers
4     print 'Learning for %s vectors'%len(self.inputvectors)
5     for trainingPhase in range(self.number_of_phase):
6       for t in range(self.iterations[trainingPhase]):
7   try:
8     k = random.choice(kv) # Random selection of one identifier
        from the identifiers list
9     kv.remove(k) # The selected identifier is removed from the
        initial list
10  except IndexError: # If the identifiers list is empty, this
      list is re-initialized to keep selecting vectors until the
      iterative process is over.
11    kv = range(len(self.inputvectors))
12    k = random.choice(kv)
13    kv.remove(k)
14  Map = Map + self.adjustment(k, t, trainingPhase, Map, self.
      findBMU(k, Map))
15    self.Map = Map
16    MapFile = open('map_%sx%s.dat' % (self.X,self.Y), 'w')
17    cPickle.dump(Map, MapFile) # Write Map into file map.dat
18    MapFile.close()
19    return self.Map
```

La présentation des vecteurs d'entrée dans un ordre aléatoire permet de retirer tout biais pouvant être introduit par l'ordre de présentation des vecteurs.

Étape 2 : Recherche du neurone vainqueur

Le neurone vainqueur est représenté par le vecteur $\Omega = \{\omega_k\}_{k=1,2,\cdots,n}$ de la carte M le plus proche en terme de distance euclidienne du vecteur d'entrée $V = \{v_k\}_{k=1,2,\cdots,n}$. Ce vecteur Ω particulier est noté *BMU* pour *Best Matching Unit*. La distance euclidienne entre un vecteur d'entrée V et

un neurone Ω_{ij} est donnée par la formule :

$$\|\Omega_{ij} - V\| = \sqrt{\sum_{k=1}^{n}(\omega_{ijk} - v_k)^2} \tag{4.3}$$

$$= \sqrt{(\Omega_{ij} - V)^T \cdot (\Omega_{ij} - V)} \tag{4.4}$$

avec $(\Omega_{ij} - V)^T$ la transposée de la matrice $\Omega_{ij} - V$.

Afin d'optimiser le code, la recherche de la distance minimale repose sur le calcul de la matrice de distances Δ donnant la distance entre le vecteur V et chaque élément Ω de M.

$$\Delta = \begin{pmatrix} \|(\Omega_{11} - V\| & \|(\Omega_{12} - V\| & \cdots & \|\Omega_{1\pi} - V\| \\ \|(\Omega_{21} - V\| & \cdots & & \\ \vdots & & & \\ \|(\Omega_{\nu 1} - V\| & & & \end{pmatrix} \tag{4.5}$$

L'élément minimal de cette matrice est ensuite recherché, sa position est notée et celle ci correspond à la position du neurone vainqueur. La position du neurone vainqueur sera notée (β_1, β_2) où β_1 est la position de la ligne dans la matrice et β_2 la position de la colonne. L'implémentation python avec l'aide du module numpy [136] pour le calcul matriciel est la suivante :

```
1  def findBMU(self, k, Map):
2      """
3        Find the Best Matching Unit for the input vector number k
4      """
5      V=numpy.ones((self.X,self.Y,self.cardinal))*self.
           inputvectors[k]
6      distmat = numpy.sum( (V - Map) * ( (V-Map)* (numpy.ones([
           self.X,self.Y,self.cardinal])*numpy.ones([1,self.cardinal
           ])) ) , axis=2 )**0.5 # Calculation of the distances
           matrix (Delta)
7      self.BMUindices = [e[0] for e in numpy.where(distmat==
           distmat.min())] # Find the position of the minimal
           element of the distances matrix
8      return self.BMUindices # Return the position of the BMU
```

Étape 3 : Règles de modification des poids

L'ajustement de la carte de Kohonen M par un vecteur V dont le neurone vainqueur correspondant se trouve à la position (β_1, β_2), est donné par la formule suivante :

$$M(t+1) = M(t) + \alpha(t) \cdot \Theta(t, \beta_1, \beta_2) \odot (V - \Omega_{ij}(t))_{1 \leq i \leq \nu, 1 \leq j \leq \pi} \tag{4.6}$$

$\odot$ représente le produit terme à terme de deux matrices ou produit de Hadamard.

$(V - \Omega_{ij}(t))_{1 \leq i \leq \nu, 1 \leq j \leq \pi}$ représente une matrice de dimension $\nu \times \pi$ composée de vecteurs de dimension n résultant de la différence entre les vecteurs V et Ω.

$\alpha(t)$ est le coefficient d'apprentissage. C'est une fonction scalaire donnée par la relation :

$$\alpha(t) = (\alpha(t_i) - \alpha(t_f)) \cdot \exp(-\frac{t}{\lambda}) + \alpha(t_f) \tag{4.7}$$

elle pondère de façon décroissante l'apprentissage au cours du temps. Pour l'implémentation développée ici la constante de temps λ est définie ainsi :

$$\lambda = \frac{t_f}{10} \tag{4.8}$$

L'implémentation python du coefficient d'apprentissage $\alpha(t)$ est donnée par le code source suivant :

```
1  def learningRate(self, t, trainingPhase=0):
2    timeCte = float(self.iterations[trainingPhase])/10
3    self.learning = ( self.alpha_begin[trainingPhase] - self.
         alpha_end[trainingPhase] ) * math.exp( -t/timeCte ) +
         self.alpha_end[trainingPhase]
4    return self.learning
```

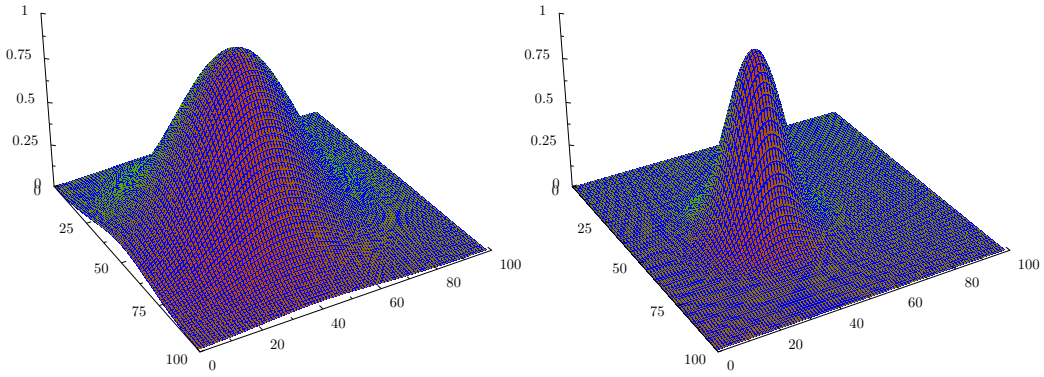
$\Theta(t, \beta_1, \beta_2)$, fonction de voisinage, est une matrice de la même dimension que M. Cette fonction dépend de l'itération t, et des coordonnées du neurone vainqueur dans la matrice M pour un vecteur V donné. La fonction de voisinage est donnée par la fonction gaussienne suivante :

$$\Theta(t, \beta_1, \beta_2) = \left(\exp\left(-\frac{\sqrt{(i - \beta_1)^2 + (j - \beta_2)^2}}{2\sigma^2(t)} \right) \right)_{1 \leq i \leq \nu, 1 \leq j \leq \pi} \tag{4.9}$$

La représentation graphique de cette fonction pour deux valeurs de rayon σ est donnée en figure 4.5 page ci-contre.

Cette fonction permet une pondération spatiale de l'apprentissage au sein de la carte. Le neurone vainqueur est celui qui est le plus influencé par le vecteur d'entrée. Les neurones voisins sont moins influencés et plus on s'éloigne du neurone vainqueur plus l'influence est faible. De plus, le rayon σ décroît au cours des itérations t. Cette décroissance est donnée par l'exponentielle décroissante :

$$\sigma(t) = (\sigma(t_i) - \sigma(t_f)) \cdot \exp(-\frac{t}{\lambda}) + \sigma(t_f) \tag{4.10}$$



(a) Rayon $\sigma = 20$ (b) Rayon $\sigma = 10$

FIGURE 4.5 – Fonction gaussienne $\Theta(t, \beta_1, \beta_2)$ pour une matrice 100×100 avec $\beta_1 = 50$ et $\beta_2 = 40$ et deux valeurs de rayon σ différentes.

La décroissance de ce rayon est représentée en figures 4.5a et 4.5b pour des rayons de 20 et 10 respectivement.

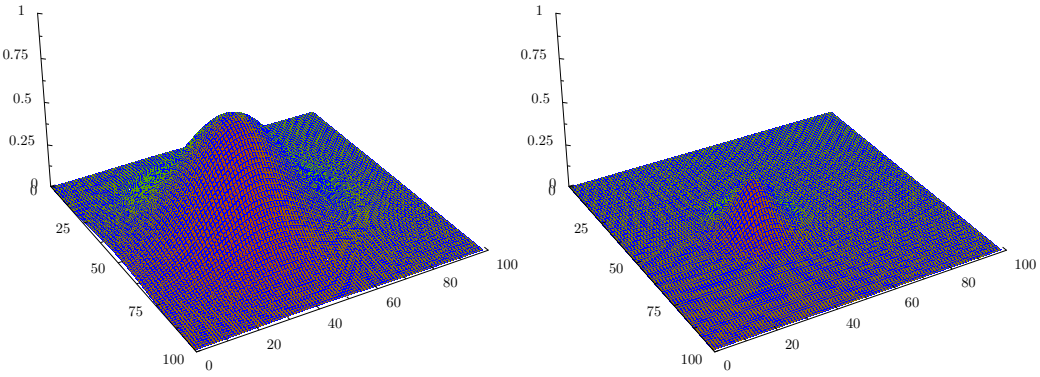
Le code source suivant montre l'implémentation python de la fonction de voisinage $\Theta(t, \beta_1, \beta_2)$ pour un élément matriciel Θ_{ij} :

```
1  def BMUneighbourhood(self, t, i, j, BMUindices, trainingPhase,
       Map):
2    dist = ( (i - BMUindices[0])**2 + (j - BMUindices[1])**2 )
       ** 0.5
3    self.neighbourhood = math.exp( - dist**2 / ( 2*(self.
       radiusFunction(t, trainingPhase)**2) ) )
4    return self.neighbourhood
```

La décroissance exponentielle du rayon σ est implémentée de la façon suivante :

```
1  def radiusFunction(self, t, trainingPhase=0):
2    timeCte = float(self.iterations[trainingPhase])/10
3    self.radius = ( self.radius_begin[trainingPhase] - self.
       radius_end[trainingPhase] ) * math.exp( -t/timeCte ) +
       self.radius_end[trainingPhase]
4    return self.radius
```

Ainsi le facteur pondérateur $\alpha(t) \cdot \Theta(t, \beta_1, \beta_2)$ permet à la fois une décroissance spatiale et temporelle de l'apprentissage comme le montre la figure 4.6 page suivante.



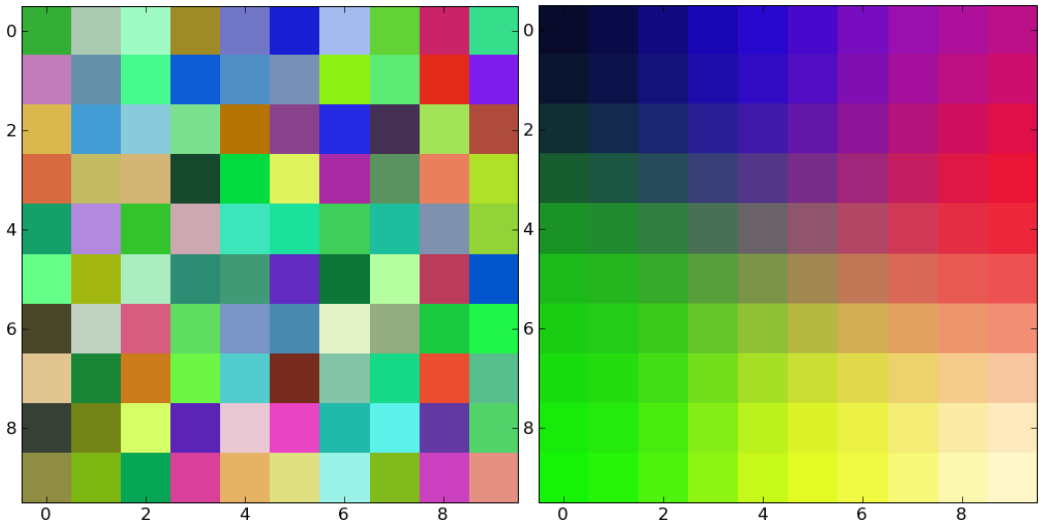
(a) Rayon $\sigma = 20$, $\alpha = 0,5$ (b) Rayon $\sigma = 10$, $\alpha = 0,25$

FIGURE 4.6 – Fonction gaussienne $\alpha(t) \cdot \Theta(t, \beta_1, \beta_2)$ pour une matrice 100×100 avec $\beta_1 = 50$ et $\beta_2 = 40$ et deux valeurs de rayon σ et de coefficient d'apprentissage α différentes.

La matrice d'ajustement $\alpha(t) \cdot \Theta(t, \beta_1, \beta_2) \odot (V - \Omega_{ij}(t))_{1 \leq i \leq \nu, 1 \leq j \leq \pi}$ est ainsi calculée :

```
1  def adjustment(self, k, t, trainingPhase, Map, BMUindices):
2    self.adjustMap = numpy.zeros((self.X, self.Y, self.cardinal)
       )
3    learning = self.learningRate(t, trainingPhase)
4    for i in range(self.X):
5      for j in range(self.Y):
6        W = Map[i,j]
7        neighbourhood = self.BMUneighbourhood(t, i, j,
           BMUindices, trainingPhase, Map)
8        self.adjustMap[i,j] = neighbourhood * learning * (self.
           inputvectors[k] - W)
9    return self.adjustMap
```

Afin de visualiser l'effet de l'apprentissage sur la carte de Kohonen nous pouvons utiliser une carte dont chaque élément code pour une couleur dans l'espace RVB (Rouge, Vert, Bleu). La couleur rouge est représentée par le vecteur $(255, 0, 0)$, la couleur verte par le vecteur $(0, 255, 0)$ et la couleur bleu par le vecteur $(0, 0, 255)$. Huit vecteurs d'entrées sont utilisés lors de l'apprentissage. La figure 4.7 page ci-contre illustre le résultat de l'apprentissage (figure 4.7b page suivante) à partir d'une carte aléatoire (figure 4.7a page ci-contre).



(a) Carte initiale (b) Carte finale

FIGURE 4.7 – Carte de Kohonen (10×10) avant et après apprentissage avec huit vecteurs d'entrée : (255,0,0), (0,255,0), (0,0,255), (255,255,255), (176,20,187), (20,198,17), (0,0,0), (255,255,0).

4.2.4 Regroupement hiérarchique de la carte de Kohonen

L'analyse des cartes de Kohonen est souvent complexe et peu facile à réaliser. De plus la visualisation des résultats sous forme d'une carte n'est pas triviale.

Nous proposons ici d'effectuer un regroupement hiérarchique de la carte de Kohonen et de représenter celle ci sous forme d'un dendrogramme ou encore plus communément appelé arbre. La représentation sous forme d'arbre de la carte de Kohonen permet d'obtenir un diagramme simple à lire et à comparer avec d'autres méthodes de *clustering*. De plus cette représentation est facile à manipuler informatiquement. L'algorithme de regroupement hiérarchique et de représentation de la carte de Kohonen sous forme d'arbre a été développé initialement par Samsonova *et al.* [153].

Schématiquement l'analyse hiérarchique de la carte de Kohonen passe par l'utilisation d'un seuil T (*threshold*) de distance. Pour un seuil T donné, deux neurones de la carte de Kohonen notés $\Omega_{i_1 j_1}$ et $\Omega_{i_2 j_2}$ sont considérés comme appartenant au même groupe si la distance euclidienne entre ces neurones ($\|\Omega_{i_1 j_1} - \Omega_{i_2 j_2}\|$) est inférieure à ce seuil T. De plus un groupe

est nécessairement un ensemble donc composé d'éléments adjacents (les neurones adjacents sont les quatre neurones directement au dessus, en dessous, à droite et à gauche du neurone considéré). La définition 1 définit la condition d'appartenance à un groupe :

Définition 1
Pour un seuil T donné, deux neurones adjacents $\Omega_{i_1 j_1}$ et $\Omega_{i_2 j_2}$ appartiennent au même groupe si et seulement si :

$$\|\Omega_{i_1 j_1} - \Omega_{i_2 j_2}\| < T \tag{4.11}$$

Pour un seuil nul le nombre de groupe est égal au nombre de neurones de la carte (chaque neurone étant différent). Pour un seuil infini il n'existe qu'un unique groupe qui contient l'ensemble des neurones de la carte. Ainsi en faisant varier le seuil T il est possible de regrouper de manière hiérarchique l'ensemble des neurones de la carte de Kohonen, la taille des groupes augmentant au fur et à mesure que le seuil T augmente jusqu'à n'obtenir qu'un unique groupe englobant toute la carte.

Afin de relier les éléments de la carte de Kohonen aux données d'entrée il est nécessaire d'associer chaque vecteur à un élément de la carte. Cette association est appelée calibration. La calibration utilise le même algorithme que la recherche du neurone vainqueur (voir page 78), c'est à dire que le vecteur d'entrée est associé au neurone le plus proche en terme de distance euclidienne.

Le regroupement hiérarchique des données étant le résultat final souhaité, les variations de seuil intéressantes sont celles qui permettent l'apparition d'un nouveau groupe de neurones au sein duquel au moins un correspond à un vecteur d'entrée. Ainsi, au cas où le nombre de neurones est supérieur au nombre de vecteurs d'entrée (cas correspondant à l'exemple donné ici), l'algorithme de regroupement hiérarchique peut s'interrompre lorsque le nombre de groupes est égal au nombre de vecteurs d'entrée (voir figure 4.8 page ci-contre). Dans le cas de l'application de cette méthode au problème du criblage virtuel le nombre de molécules donc de vecteurs d'entrée est supérieur au nombre de neurones. Ainsi, dans ce cas précis plusieurs vecteurs excitent un même neurone (c'est à dire que plusieurs vecteurs sont calibrés sur le même neurone) et tous les neurones sont impliqués par la calibration, donc l'algorithme de regroupement hiérarchique s'arrête lorsque le nombre de groupes formés est égal au nombre de neurones dans la carte.

La combinaison de l'algorithme d'analyse hiérarchique (voir algorithme 3 page 86) avec la calibration des données d'entrée sur la carte permet d'obtenir le dendrogramme présenté en figure 4.9 page ci-contre.

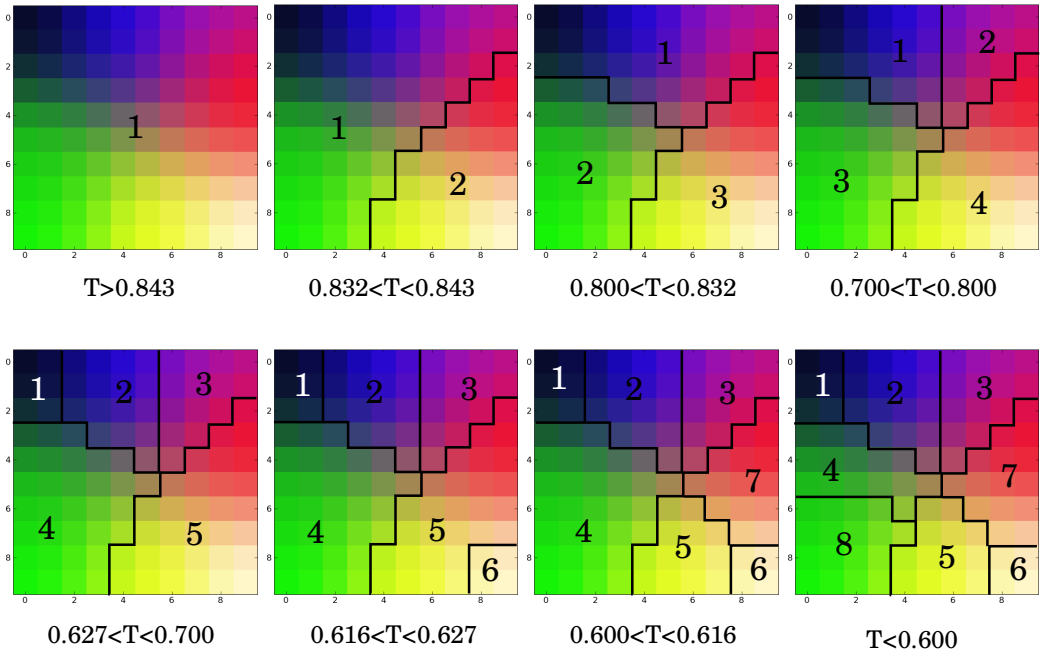


FIGURE 4.8 – Ensemble des groupes formés lors de l'algorithme d'analyse hiérarchique de la carte de Kohonen. Le seuil T considéré pour chaque carte est indiqué en dessous de celle-ci.

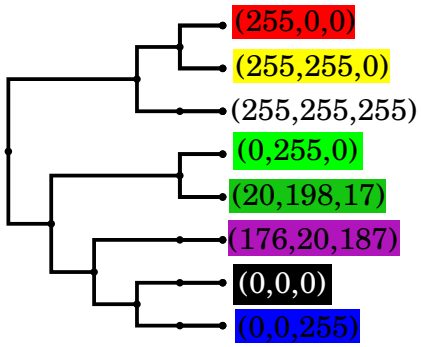


FIGURE 4.9 – Dendrogramme issu du regroupement hiérarchique des vecteurs d'entrée calibrés sur la carte de Kohonen présenté en figure 4.7b page 83

L'algorithme permettant l'analyse hiérarchique non supervisée des cartes de Kohonen est le suivant :

Algorithme 3 (Regroupement hiérarchique de la carte de Kohonen) *Cet algorithme repose sur deux procédures interdépendantes :*
Procédure de nucléation (*cluster-discovery*) pour un seuil de distance T
- *Marquer l'ensemble des neurones comme non visités (neurone N)*
- *Tant qu'il y a des neurones non visités :*
  - *Localiser un neurone arbitraire N*
  - *Générer un nouveau centre de nucléation C*
  - *Appeler la procédure de regroupement pour N, C et T*

Procédure de regroupement (*cluster*) pour N, C et T
- *Attribuer N comme C*
- *Marquer N comme visité*
- *Pour chaque neurone A non visité et adjacent à N tel que la distance $|NA| < T$ appeler la procédure de regroupement pour A, C et T*

L'implémentation python de l'algorithme 3 est présentée ci dessous :

```
1   def clusterDiscovery(self,Map,T): #Procedure Cluster-Discovery
2     borderMat = self.borderline(Map)
3     Nv = [[i,j] for i in range(self.X) for j in range(self.Y)] #
          Mark all nodes as unvisited
4     random.shuffle(Nv)
5     Cv = []
6     while Nv != []: #While there are unvisited nodes
7       N = Nv[0] # locate an arbitrary node N
8       C = [] # start a new cluster C
9       A = [] # Adjacent nodes
10      A, C, Nv = self.cluster(borderMat,Nv,N,T,C,A) # call
            procedure cluster for N, C and T
11      while A != []:
12        N = A[0]
13        A.remove(N)
14        A, C, Nv = self.cluster(borderMat,Nv,N,T,C,A) # for
              each unvisited node A adjacent to N call procedure
              cluster for N, C and T
15      Cv.append(C)
16    clustersMap = numpy.zeros((self.X,self.Y))
17    CId = 0
18    for v in Cv:
19      CId = CId + 1
20      for e in v:
21        clustersMap[e[0],e[1]] = CId
```

```
22      self.clustersMap = clustersMap
23      return self.clustersMap
24
25    def cluster(self,borderMat,Nv,N,T,C,A):
26      C.append(N) # assign N to C
27      #print 'Nv%s'%Nv
28      Nv.remove(N) # mark N as visited
29      c = 0
30      for i in range(N[0]*2,N[0]*2+2):
31        for j in range(N[1]*2,N[1]*2+2):
32            c = c + 1
33            if borderMat[i,j] <= T: # distance < Threshold
34              if (c == 1 and N[1] >= 1):
35                fc = [N[0],N[1]-1]
36                if (fc in Nv and fc not in A): # unvisited node
37                  A.append(fc)
38              if (c == 2 and N[0] >= 1):
39                fc = [N[0]-1,N[1]]
40                if (fc in Nv and fc not in A): # unvisited node
41                  A.append(fc)
42              if (c == 3 and N[0] <= self.X - 1):
43                fc = [N[0]+1,N[1]]
44                if (fc in Nv and fc not in A): # unvisited node
45                  A.append(fc)
46              if (c == 4 and N[1] <= self.Y - 1):
47                fc = [N[0],N[1]+1]
48                if (fc in Nv and fc not in A): # unvisited node
49                  A.append(fc)
50      return A, C, Nv
```

4.2.5 Une autre méthode de regroupement : *k-means clustering*

L'algorithme *k-means*

La méthode des k-moyennes, plus communément appelée la méthode *k-means* [103, 76] permet de classer en k groupes des vecteurs suivant la moyenne des éléments du groupe. Cette méthode, plus ancienne et plus classique que les réseaux de Kohonen, a aussi été implémenté dans le logiciel AuPosSOM afin d'en effectuer la comparaison. L'étape de génération des vecteurs à partir des poses de *docking* n'est pas modifié mais juste la manière de regrouper les vecteurs est différente.

Soit un ensemble de vecteurs d'entrée $\{V_j\}$. L'algorithme *k-means* permet de former k ensembles $\mathbb{E}_i$ qui minimisent la somme des distances euclidiennes (voir équation 4.3 page 79) entre chaque élément de l'ensemble et la moyenne

μ_i des éléments de l'ensemble – encore appelée centroïde. L'algorithme revient à rechercher les ensembles $\{\mathbb{E}_i\}_{i=1,2,\cdots,k}$ qui minimisent la fonction :

$$\chi = \sum_{i=1}^{k} \sum_{V_j \in \mathbb{E}_i} \|V_j - \mu_i\| \tag{4.12}$$

L'algorithme 4 propose une méthode itérative de calcul des k groupes minimisant la fonction χ.

Algorithme 4
Initiation : k vecteurs V sont aléatoirement choisis comme centroïdes μ_i, et initiation de la variable μ_i^{old} :

$$\mu_i^{old} \leftarrow (0, 0, \cdots, 0) \forall i = \{1, 2, \cdots, k\} \tag{4.13}$$

Tant que $\exists i \in \{1, 2, \cdots, k\}$ *tel que* $\mu_i^{old} \neq \mu_i$ ***:***

$$\mu_i^{old} \leftarrow \mu_i \forall i = \{1, 2, \cdots, k\} \tag{4.14}$$

Étape d'attribution : Les k ensembles $\{\mathbb{E}_i\}_{i=1,2,\cdots,k}$ sont définis ainsi :

$$\mathbb{E}_i \leftarrow \{V_j : \|V_j - \mu_i\| \leq \|V_j - \mu_{i*}\| \forall i* = \{1, 2, \cdots, k\}\} \tag{4.15}$$

Étape d'ajustement : la position du centroïde μ_i de chaque ensemble $\mathbb{E}_i$ est ajustée suivant la relation (moyenne arithmétique des vecteurs V de $\mathbb{E}_i$) :

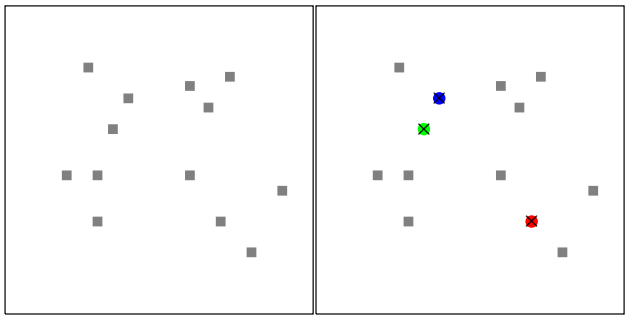
$$\mu_i \leftarrow \frac{1}{card(\mathbb{E}_i)} \sum_{c=1}^{card(V)} \sum_{j=1}^{card(\mathbb{E}_i)} v_{c,j} \tag{4.16}$$

$v_{c,j}$ étant l'élément c du vecteur V_j.

L'implémentation python de l'algorithme *k-means* est présentée en page 231. La figure 4.10 page ci-contre illustre le fonctionnement de l'algorithme 4.

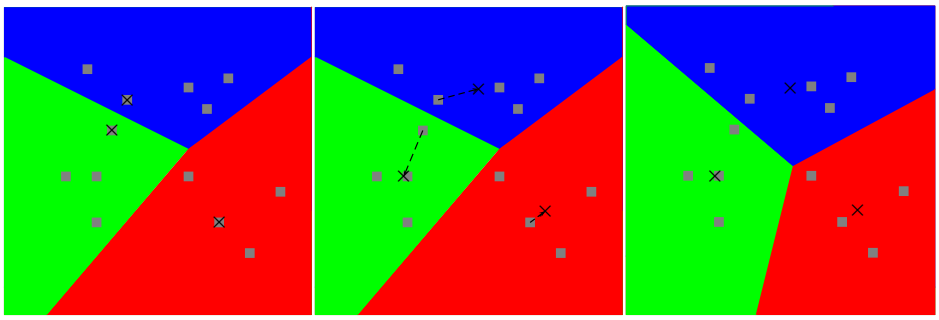
L'algorithme UPGMA

Afin de représenter sous forme d'arbre le regroupement effectuer par la méthode *k-means*, l'algorithme UPGMA (*Unweighted Pair Group Method with Arithmetic Mean*) [160] a été utilisé. Cette méthode permet de représenter sous forme d'arbre la distance entre les centroïdes, l'ensemble des molécules appartenant à un *cluster* étant assimilé au centroïde de ce *cluster*. La construction de l'arbre passe alors par l'analyse d'une matrice de distances euclidiennes entre centroïdes. Les deux centroïdes les plus proches sont regroupés en un ensemble. Les distances entre ce nouvel ensemble et les



(a) Ensemble des vecteurs, ici à deux dimensions, représenté dans le plan.

(b) Sélection aléatoire de $k = 3$ vecteurs comme centroïdes μ.



(c) Création des k ensembles suivant l'équation 4.15 page ci-contre.

(d) Ajustement des centroïdes selon l'équation 4.16 page précédente.

(e) Les étapes (c) et (d) sont répétées jusqu'à convergence.

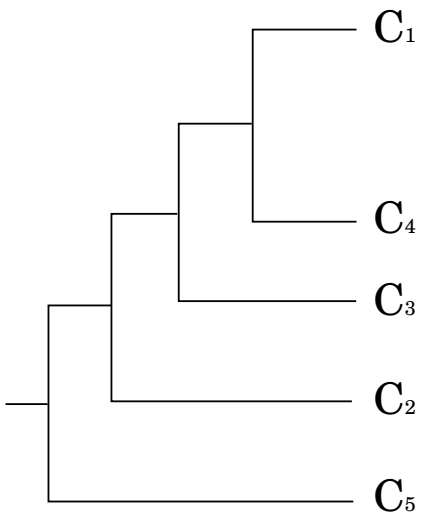
FIGURE 4.10 – Algorithme *k-means* pour des vecteurs à deux dimensions et $k = 3$.

autres centroïdes sont alors calculés comme la moyenne des distances entre le centroïde considéré et les deux centroïdes regroupés. De manière itérative, la topologie de l'ensemble de l'arbre est alors calculée. Un exemple de calcul utilisant l'algorithme UPGMA sur une matrice de distance 5×5 est donné en figure 4.11 page ci-contre.

$$\left(\begin{array}{c|ccccc} & C_1 & C_2 & C_3 & C_4 & C_5 \\ \hline C_1 & 0 & \cdot & \cdot & \cdot & \cdot \\ C_2 & 5 & 0 & \cdot & \cdot & \cdot \\ C_3 & 3 & 7 & 0 & \cdot & \cdot \\ C_4 & 1 & 5 & 5 & 0 & \cdot \\ C_5 & 12 & 5 & 8 & 12 & 0 \end{array}\right) \rightarrow \left(\begin{array}{c|cccc} & C_{1,4} & C_2 & C_3 & C_5 \\ \hline C_{1,4} & 0 & \cdot & \cdot & \cdot \\ C_2 & 5 & 0 & \cdot & \cdot \\ C_3 & 4 & 7 & 0 & \cdot \\ C_5 & 12 & 5 & 8 & 0 \end{array}\right)$$

$$\left(\begin{array}{c|ccc} & C_2 & C_{3,1,4} & C_5 \\ \hline C_2 & 0 & \cdot & \cdot \\ C_{3,1,4} & 6 & 0 & \cdot \\ C_5 & 5 & 10 & 0 \end{array}\right)$$

(a) Exemple de calcul utilisant l'algorithme UPGMA sur une matrice de distance 5×5.



(b) Arbre correspondant à la matrice de distance ci dessus. Seule la topologie de l'arbre est indiqué, les distances sont ignorées.

FIGURE 4.11 – Application de l'algorithme UPGMA

Chapitre 5

Résonance Magnétique Nucléaire

> In theory there is no difference between theory and practice, but in practice there is.
>
> Jan L. A. van de Snepscheut

Afin d'étudier les interactions faibles entre un ligand et une macromolécule biologique, la Résonance Magnétique Nucléaire (RMN) est une technique de choix. Elle permet premièrement de mettre en évidence une interaction. Une analyse plus poussée des résultats permet aussi de définir précisément, à l'échelle atomique, quels sont les atomes les plus impliqués dans l'interaction. La RMN permet aussi de quantifier la force de l'interaction par la détermination de la constante d'équilibre. Cette technique spectroscopique permet aussi d'étudier les interactions de plusieurs ligands au sein d'un même site. Ce phénomène de compétition peut également être quantifié.

Le processus de liaison d'une petite molécule organique à une macromolécule biologique entraîne des modifications importantes des propriétés physiques (la dynamique du ligand, son environnement). Les méthodes RMN permettant la détection de la liaison d'une molécule à une autre, souvent de taille plus importante, reposent toutes sur la détection de ces modifications des propriétés physiques du ligand par des études de diffusion, de changement de régime de relaxation ou par effet NOE (*Nuclear Overhauser Effect*).

Le but de ce chapitre est de décrire du point de vue théorique et pratique les expériences RMN qui permettent de mettre en évidence et d'étudier les interactions entre une macromolécule biologique – une protéine – et une petite molécule organique – le ligand.

5.1 Modèle théorique d'interaction à deux états

Les interactions récepteur (R) ligands (L) reposent sur des interactions faibles entre atomes. Ces interactions réversibles reposent classiquement sur un équilibre entre deux états : un état lié $L \cdot R$ et un état libre :

$$L + R \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} L \cdot R$$

k_1 et k_{-1} sont les constantes cinétiques d'association et de dissociation, respectivement. Par définition, à l'équilibre la vitesse de formation du complexe $L \cdot R$ est égale à la vitesse de dissociation de ces deux entités. Cette équilibre peut être caractérisé par une constante de dissociation notée K_d et exprimée par la relation :

$$K_d = \frac{[L][R]}{[L \cdot R]} = \frac{k_{-1}}{k_1} \tag{5.1}$$

La vitesse d'association est essentiellement limitée par la diffusion (k_1 est de l'ordre de $10^7 s^{-1} M^{-1}$) mais peut être réduit dans le cas d'un réarrangement conformationnel important du ligand et/ou du récepteur. À concentration constante en récepteur nous pouvons définir k_1' constante cinétique de pseudo premier ordre :

$$k_1' = k_1[R] \tag{5.2}$$

Du fait de l'équilibre entre les deux états libres et liés, le ligand coexiste en solution sous deux formes. Les fractions molaires liées χ_b et libres χ_f peuvent ainsi être définies :

$$\chi_b = \frac{[L \cdot R]}{[L]_0} = 1 - \chi_f \tag{5.3}$$

avec $[L]_0$ la concentration en ligand initialement mis en solution.

Il est primordial de comparer les constantes de vitesses (k_1 et k_{-1}) caractérisant l'équilibre entre l'état libre (f pour *free*) et l'état lié (b pour *bounded*) et les échelles de temps qui régissent les phénomènes mis en évidence par la RMN. Dans un premier temps, nous pouvons considérer trois échelles de temps en RMN :

- l'échelle du déplacement chimique, avec les fréquences de résonance ν^f et ν^b correspondant respectivement aux fréquences de résonance de la forme libre et de la forme liée du ligand.
- l'échelle de la relaxation longitudinale caractérisée par le temps caractéristique $T_{1,I}$ pour un noyau de spin I. T_1 détermine la fréquence minimale des phénomènes que l'on peut mettre en évidence au cours d'une expérience RMN.

	Échelle du déplacement chimique	Échelle de la relaxation longitudinale $1/T_1$ du ligand libre	Échelle de la relaxation croisée
Condition d'échange rapide	$k_{-1} + k'_1 \gg 2\pi\lvert\nu_I^f - \nu_I^b\rvert$	$k_{-1} + k'_1 \gg 1/T_{1,I}^f$	$k_{-1} + k'_1 \gg \lvert\sigma_{IS}^b\rvert + \lvert\sigma_{IS}^f\rvert$

TABLE 5.1 – Condition d'échange rapide suivant les trois échelles de temps de la RMN. La relation $a \gg b$, signifie ici : $a \geq 10b$.

– l'échelle de temps de la relaxation croisée reliée à la vitesse de relaxation croisée σ_{IS}^b et σ_{IS}^f pour l'état lié et libre du ligand, respectivement.

Dans le cas de l'étude RMN d'un système en échange entre deux états, il est nécessaire de préciser les conditions dites de l'échange rapide et ceci pour les trois échelles de temps définies ci-dessus. Le tableau 5.1 décrit les conditions de l'échange rapide en fonction de l'échelle de temps RMN considéré. Les conditions d'échange rapide se manifeste par les phénomènes suivant.

Un échange rapide relativement à l'échelle du déplacement chimique des états libres et liés conduit à une moyenne des déplacements chimiques pondérée par les fractions des états libres et liés du ligand :

$$\delta_{obs} = \chi_f \delta_f + \chi_b \delta_b \tag{5.4}$$

où δ_{obs} est le déplacement chimique observé et δ_f, δ_b les déplacements chimiques des états libres et liés respectivement.

Un échange rapide par rapport à l'échelle de la relaxation longitudinale permet d'observer sur la forme libre les caractéristiques physiques de l'état lié du ligand.

Finalement, un échange rapide par rapport à l'échelle de la relaxation croisée permet d'observer des effets NOE [1] sur la forme libre provenant majoritairement de la forme liée – on parle alors de NOE transféré.

Les études d'interaction protéine·ligand présentées ici sont basés sur la détection des signaux du ligand. L'échange rapide entre l'état lié et l'état libre par rapport à l'échelle de relaxation longitudinale permet d'observer sur la forme libre les propriétés physiques de la forme liée. Ainsi le ligand libre conserve une mémoire de l'état lié : c'est sur ce phénomène de transfert qu'est basée la détection d'une molécule ligand par rapport à une molécule non-ligand. On parle souvent de *transfer NMR*. Les signaux fins observés dans ces expériences sont effectivement ceux de la forme libre du ligand.

1. *Nuclear Overhauser Effect*

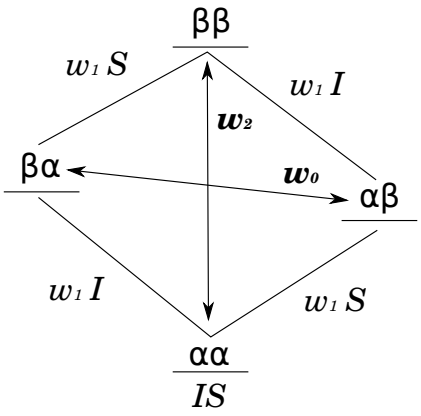


FIGURE 5.1 – Niveaux d'énergies d'une paire de spins I et S et les six chemins possibles de relaxation avec les probabilités de transition w associées. Les chemins de relaxation croisée correspondant aux transitions zéro quantum w_0 et double quanta w_2 donnent lieu à l'effet NOE.

5.2 L'effet NOE

L'effet NOE (*Nuclear Overhauser Effect*) est à la base même de l'ensemble des expériences RMN décrites ici. Il résulte de l'interaction dipolaire entre plusieurs spins proche dans l'espace. Cet effet est donc important afin de déterminer la structure tridimensionnelle des molécules, puisqu'il permet la mesure de distances entre noyaux.

L'effet NOE peut être mis en évidence par la saturation d'un spin S par l'application d'un champ de radiofréquence à la fréquence de résonance de S. Le premier effet de la saturation est la disparition du signal de S. Le deuxième effet est la possible variation de l'intensité du signal de I, si les deux spins I et S présentent une interaction dipolaire non négligeable.

Afin de comprendre l'origine de l'effet NOE, il est nécessaire de détailler les différents chemins de relaxation d'une paire de spins I et S (voir figure 5.1). Seules les transitions simultanées de deux spins, phénomène de relaxation croisée correspondant aux transitions zéro quantum w_0 et double quanta w_2, sont à l'origine des transferts d'aimantation donnant lieu à l'effet NOE.

Le traitement quantitatif de l'effet NOE permet de donner l'expression de l'accroissement du signal de I par saturation de S :

$$NOE = 1 + \underbrace{\frac{w_2^{IS} - w_0^{IS}}{2w_1^S + w_2^{IS} + w_0^{IS}} \cdot \frac{\gamma_S}{\gamma_I}}_{\eta} \tag{5.5}$$

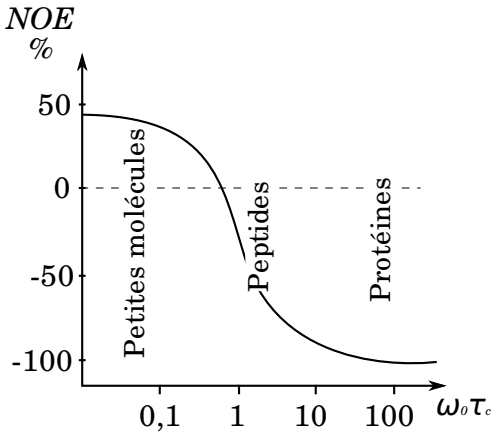


FIGURE 5.2 – Variation de l'intensité de l'effet NOE en fonction de $\omega_0\tau_c$ (ω_0 est la vitesse angulaire pour le proton ^1H pour un spectromètre à 500MHz et τ_c est le temps de corrélation).

avec γ le rapport gyromagnétique et w les probabilités de transition entre les états considérés. η est le facteur d'amplification de l'effet NOE. Pour deux spins I et S, l'amplitude maximale de l'effet NOE est reliée à la vitesse de relaxation croisée σ_{IS} entre I et S, à la relaxation propre ρ_I et aux rapports gyromagnétiques γ_I et γ_S des noyaux I et S, respectivement :

$$\eta_{max} = \frac{\sigma_{IS}}{\rho_I} \cdot \frac{\gamma_S}{\gamma_I} \tag{5.6}$$

Le signe de l'amplitude de l'effet NOE (η) dépend du signe de $w_2^{IS} - w_0^{IS}$ donc de la transition qui prédomine pour le régime de relaxation étudié. Pour les petites molécules $w_2^{IS} > w_0^{IS}$ ainsi l'amplitude de l'effet NOE est positive. Pour les macromolécules $w_2^{IS} < w_0^{IS}$ donc l'amplitude de l'effet NOE est négative. Si on reporte l'amplitude de l'effet NOE en fonction de $\omega_0\tau_c$ (figure 5.2) où $\omega_0/(2\pi)$ est la fréquence de Larmor du noyau considéré et τ_c est le temps de corrélation de la molécule on obtient une courbe d'allure sigmoïdale qui passe par 0 en un point.

Le temps de corrélation τ_c de la molécule dépend du volume V de la molécule, de la viscosité du milieu η et de la température T :

$$\tau_c = \frac{\eta V}{k_B T} \tag{5.7}$$

La répartition $J(\omega)$ des fréquences de rotation $\omega/(2\pi)$ des molécules en so-

lution suit la lorentzienne :

$$J(\omega) = \frac{2\tau_c}{1 + (\omega\tau_c)^2} \tag{5.8}$$

D'après l'équation 5.6 page précédente l'amplitude de l'effet NOE dépend de la vitesse du phénomène de relaxation croisée σ_{IS} qui est donnée par la relation :

$$\sigma_{IS} = \left(\frac{\mu_0}{4\pi}\right)^2 \frac{\gamma_I^2\gamma_S^2\hbar^2}{4r_{IS}^6}(6J(\omega_I + \omega_S) - J(\omega_I - \omega_S)) \tag{5.9}$$

Ainsi l'efficacité avec laquelle l'aimantation est transférée d'un spin à l'autre par NOE est fortement liée à la distance entre les spins des deux noyaux considérés. Cette dépendance est en $1/r^6$ comme le montre l'équation 5.9.

Pour un système en échange chimique entre un état libre et un état lié, le ligand peut retourner à l'état libre en solution, où il conserve pendant un certain temps, l'aimantation de l'état lié au récepteur. Il est fréquent de parler de "mémoire de l'état lié" pour caractériser ce phénomène. Les effets NOEs observés sont appelés NOEs transférés ou TRNOEs [9, 10, 11, 12]. Les conditions favorables à l'obtention des TRNOEs sont les suivantes :

- Il faut se trouver dans des conditions d'échange rapide par rapport à la vitesse de relaxation du ligand libre (ρ_I^f), typiquement :

  $$k_{-1} + k_1' > 10\rho_I^f \tag{5.10}$$

  Ainsi le ligand est caractérisé par une matrice de relaxation croisée moyenne $< \sigma_{IS} >$ pondérée par les fractions molaires de la forme libre et de la forme liée du ligand :

  $$< \sigma_{IS} >= \chi_f\sigma_{IS}^f + \chi_b\sigma_{IS}^b \tag{5.11}$$

  avec χ_f et χ_b les fractions molaires respectivement libres et liées du ligand (voir équation 5.3 page 94) et σ_{IS}^f, σ_{IS}^b, les matrices de relaxation croisée de la forme libre et de la forme liée du ligand, respectivement.
- La contribution de la forme liée du ligand doit-être prépondérante afin d'observer sur la forme libre les caractéristiques de la forme liée :

  $$|\chi_b\sigma_{IS}^b| \gg |\chi_f\sigma_{IS}^f| \tag{5.12}$$

  Comme la relaxation croisée est nettement plus efficace dans une macromolécule qu'en solution ($\sigma_{IS}^b > \sigma_{IS}^f$), il est possible de travailler en large excès de ligand ce qui facilite sa détection ($\chi_f > \chi_b$).

5.3 Études d'interaction par RMN

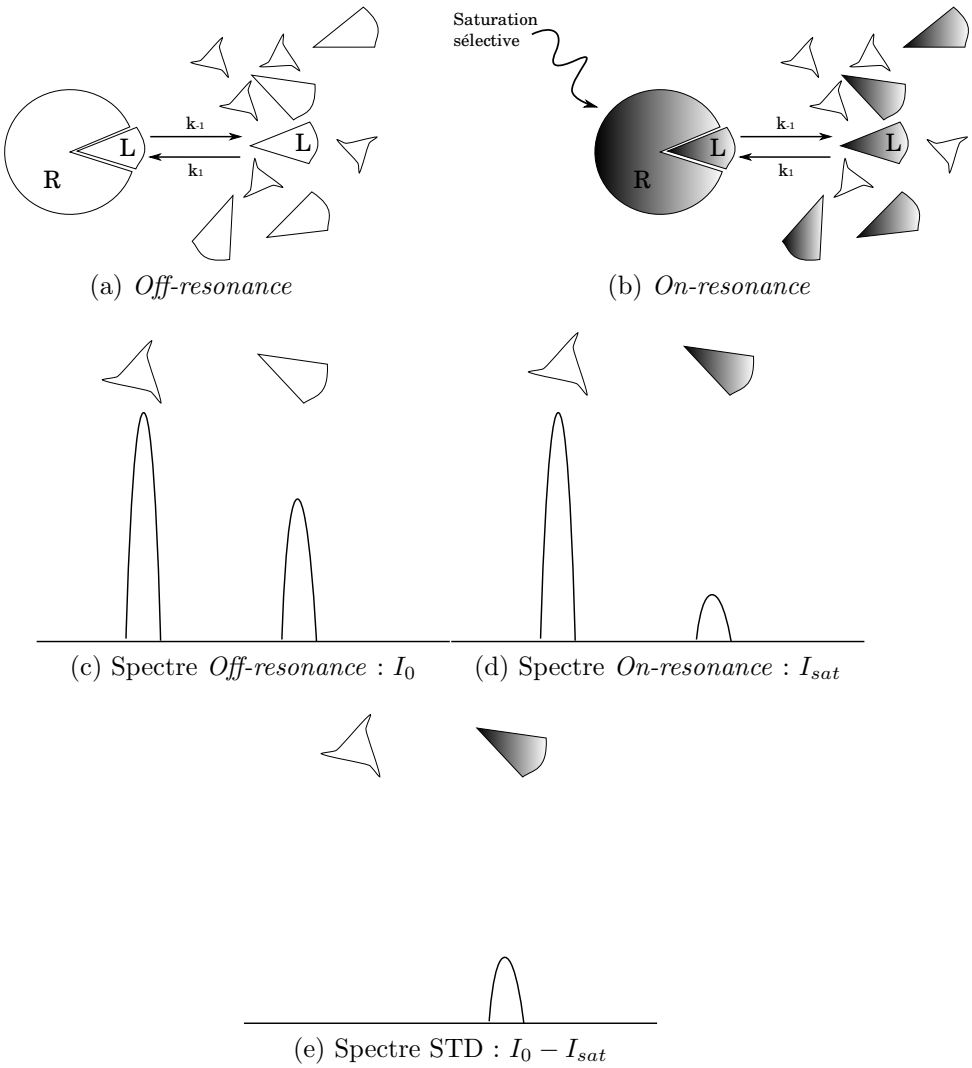
5.3.1 Différence de transfert de saturation

L'expérience STD (*Saturation Transfer Difference*) est basée sur le transfert de saturation d'une macromolécule à un ligand lié s'échangeant, par échange chimique, avec la forme libre. Les expériences basées sur le phénomène de transfert de saturation sont développées depuis les années 1960 [49] et ont repris un nouvel essor avec l'utilisation de la RMN comme moyen d'étude des interactions ligand·macromolécule [115]. Les signaux du ligand étant ceux observés dans un spectre STD il est nécessaire que la concentration en ligand libre soit détectable dans un spectre RMN à une dimension classique. Cette condition est remplie pour une gamme d'affinité allant de $K_d = 10^{-3}\,\mathrm{M}$ à $K_d = 10^{-8}\,\mathrm{M}$ [114]. Le principe de la détection des molécules ligands repose sur la saturation sélective des protons du récepteur et sa propagation par diffusion de spins au sein du récepteur – transfert intramoléculaire, très efficace pour les macromolécules – jusqu'aux protons du ligand – transfert intermoléculaire (figure 5.3b page suivante). En pratique, des spectres sont enregistrés avec (figure 5.3d page suivante) (spectre *on-resonance*) et sans saturation du récepteur (figure 5.3a page suivante et 5.3c page suivante) (spectre *off-resonance*). Par soustraction de ces deux spectres, le spectre STD est ainsi obtenu (figure 5.3e page suivante). De plus, les protons du ligand les plus proches de la protéine recevront plus de saturation comparativement à des protons plus éloignés. Ainsi, les protons les plus enfouis au sein du récepteur présenteront un signal STD plus important que les protons plus exposés au solvant. Il est donc possible à partir des intensités de déterminer la ou les zones du ligand les plus impliquées dans l'interaction avec la protéine – on parle couramment d'*epitope mapping*. Ces informations sont évidemment précieuses pour le pharmacologiste moléculaire souhaitant améliorer l'affinité d'un ligand. Le transfert de saturation reposant sur l'effet NOE, l'efficacité de transfert de saturation est dépendante en $1/r^6$ vis à vis de la distance entre les protons du récepteur et du ligand (équation 5.9 page ci-contre).

De nombreux paramètres expérimentaux vont affecter l'efficacité du transfert de saturation.

L'efficacité du transfert va augmenter avec la masse moléculaire de la macromolécule. Cette propriété est à relier directement à l'augmentation du temps de corrélation τ_c avec la masse moléculaire, ce qui augmente l'amplitude absolue du NOE dans la gamme des macromolécules (voir figure 5.2 page 97).

Concernant le ligand, celui ci doit être de petite taille vis à vis de la



(a) *Off-resonance* (b) *On-resonance*

(c) Spectre *Off-resonance* : I_0 (d) Spectre *On-resonance* : I_{sat}

(e) Spectre STD : $I_0 - I_{sat}$

FIGURE 5.3 – Principe de l'expérience de différence de transfert de saturation (STD). Le spectre *off-resonance* (c) correspond à un spectre 1D obtenu sans saturation du récepteur (a). Le spectre *on-resonance* (d) est obtenu par saturation sélective des protons du récepteur (b). Le spectre STD (e) résulte la différence du spectre *off-resonance* (c) et du spectre *on-resonance* (d).

macromolécule. Dans le cas de petites molécules à activité pharmacologique (*hit* ou *lead*) cela ne pose pas de problème, la taille du ligand est assez réduite ce qui minimise fortement la diffusion de spin au sein du ligand. Ceci permet d'obtenir des intensités STD différentes entre les protons et donc de déterminer un *epitope mapping* du ligand étudié.

Le temps de résidence du ligand au sein du site de fixation de la protéine joue aussi un rôle essentiel dans l'expérience STD. Une constante de vitesse k_{-1} élevée facilite la détection du ligand en solution. Un échange rapide entre l'état libre et l'état lié permet d'augmenter la proportion de ligand saturé en solution du fait d'un renouvellement rapide du ligand lié. Un large excès de ligand par rapport à la protéine permet aussi de faciliter la détection des ligands en STD (pour nos études nous travaillons avec un rapport $[L]/[P]$ compris entre 50 et 200).

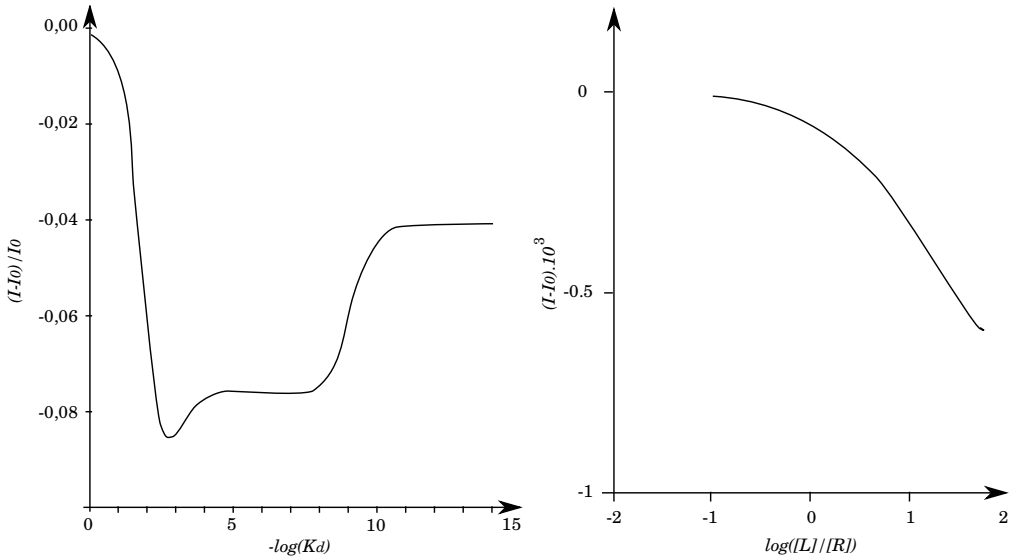
Il est possible de calculer théoriquement l'intensité du signal STD d'un proton en tenant compte des protons voisins du ligand et du récepteur, en calculant la matrice de relaxation complète (théorie CORCEMA *Complete Relaxation and Conformational Exchange Matrix*) [80]. Ainsi grâce à cette simulation, nous pouvons prévoir l'évolution de l'intensité STD en fonction de divers paramètres comme la constante d'équilibre $K_a = 1/K_d$ et le rapport $[L]/[R]$ (figure 5.4 page suivante).

La séquence d'impulsion (figure 5.5 page suivante) utilisée contient un train d'impulsions de 40 gaussiennes de 50ms séparées par un délai de 1ms afin de pouvoir saturer sélectivement les protons du récepteur. La séquence *excitation-sculpting* est utilisée afin d'éliminer le signal de l'eau [74].

5.3.2 WaterLOGSY

L'expérience WaterLOGSY (*Water-Ligand Observed via Gradient Spectroscopy*) repose sur le transfert d'aimantation entre les molécules d'eau, le récepteur et les molécules ligands [32, 31]. Comme toutes les expériences de *transfer-NMR* l'expérience WaterLOGSY repose sur l'effet NOE et donc sur la relaxation croisée. Les conditions favorables à l'obtention de NOE transférés (TRNOEs) entre l'eau le ligand et la protéine sont décrites dans le paragraphe 5.2 page 96.

La détection des molécules ligands repose sur la différence du signe de l'amplitude du NOE. La fixation d'une petite molécule sur une macromolécule va modifier son temps de corrélation τ_c et donc l'amplitude du NOE (voir figure 5.2 page 97). Le transfert NOE entre le récepteur macromoléculaire et la molécule ligand va être de signe négatif. De même, les molécules d'eau potentiellement présentent au niveau des sites de liaison des ligands s'ordonnent autour des groupements hydrophobes de ces sites. Ces cages d'eau



(a) Évolution de l'intensité STD en fonction de K_d.

(b) Évolution de l'intensité STD en fonction du rapport des concentrations en ligand et récepteur.

FIGURE 5.4 – Effet du K_d et du rapport $[L]/[R]$ sur l'intensité du signal STD pour un temps de saturation de 5 secondes.

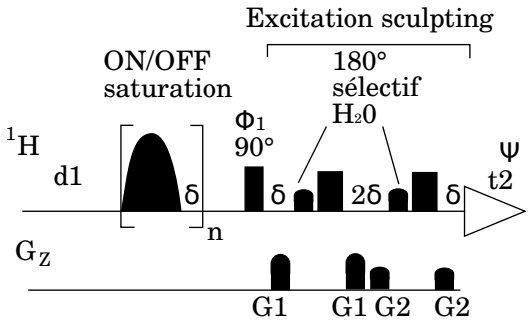


FIGURE 5.5 – Séquence RMN d'impulsions de l'expérience 1D STD ^{1}H.

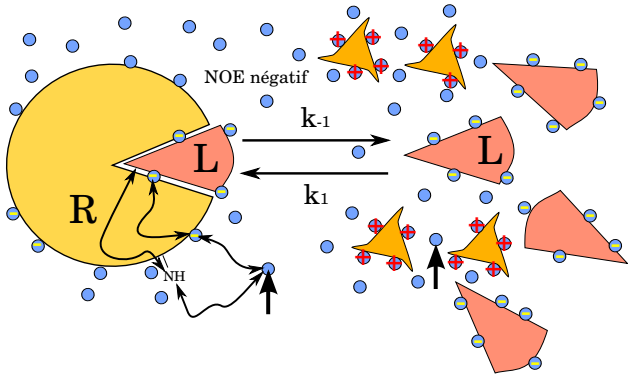


FIGURE 5.6 – Principe de l'expérience WaterLOGSY. Les flèches ↑ indiquent l'inversion sélective des spins des noyaux ^{1}H des molécules d'eau (points bleus). Les signes "−" indiquent des NOEs de signes négatifs, les signes "+" des NOEs de signes positifs.

– ou clathrate – sont caractérisées par un temps de corrélation de quelques nanosecondes comparable à celui de la macromolécule qui les porte. Ainsi les NOEs entre ces molécules d'eau et la molécule ligand sont aussi de signe négatif. Au contraire les molécules non-ligands en solution ont un temps de corrélation plus court donc un NOE de signe positif ou proche de zéro. C'est cette différence de dynamique qui est exploitée dans les expériences WaterLOGSY (figure 5.6).

La séquence d'impulsions de l'expérience WaterLOGSY (figure 5.7 page suivante) se décompose en trois étapes :

- inversion sélective des spins des noyaux ^{1}H des molécules d'eau,
- transfert de l'aimantation par NOE vers les molécules voisines pendant un temps de mélange τ_m,
- élimination du signal de l'eau, ici par *excitation-sculpting* [74].

Le spectre WaterLOGSY est classiquement phasé de telle sorte que les molécules ligands présentant des NOEs négatifs possèdent des signaux d'intensités positives et les molécules non-ligands présentant des NOEs positifs possèdent des signaux d'intensité négatives. Ainsi les molécules ligands et non-ligands sont de signe inverse sur le spectre obtenu.

Initialement, l'expérience WaterLOGSY semble limitée à l'identification de molécules ligands parmi des molécules non-ligands. Cependant, il semble que cette expérience peut aussi permettre d'étudier l'accessibilité au solvant de la molécule ligand. Cette approche a été dénommée SALMON : *Solvent Accessibility and protein Ligand binding studied by NMR Spectroscopy* [106, 105]. Afin de comparer l'*epitope mapping* obtenu par la méthode STD avec

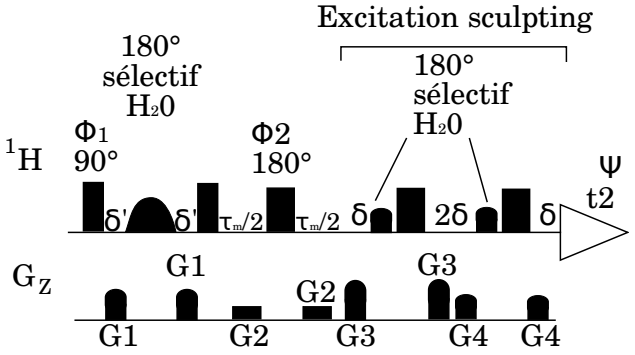


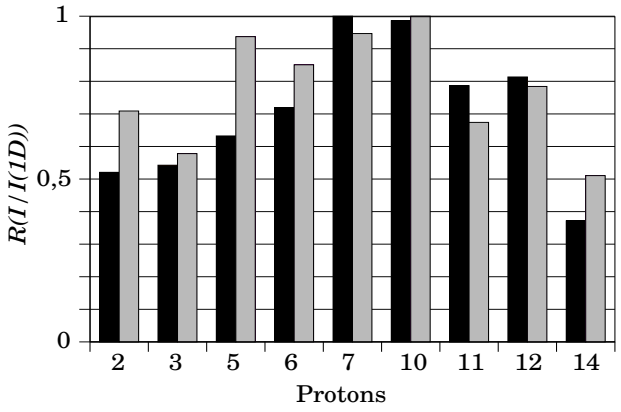
FIGURE 5.7 – Séquence RMN d'impulsions de l'expérience 1D WaterLOGSY ^{1}H.

celui obtenu par l'expérience WaterLOGSY nous avons testé ces techniques sur le modèle HSA (*Human Serum Albumin*) - naproxène, un ligand de la HSA [163]. La figure 5.8 page ci-contre montre une comparaison de l'*epitope mapping* obtenu pour le naproxène en interaction avec la HSA pour les deux méthodes : STD et WaterLOGSY. Le coefficient de corrélation entre les deux jeux de données (STD et WaterLOGSY) est de 0,78.

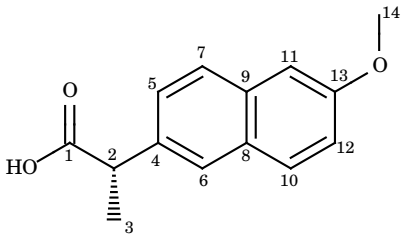
5.3.3 *Reverse NOE Pumping*

La *Reverse NOE Pumping* (RNP) [21] est une autre technique de détection de molécules en interaction avec une macromolécule. Cette technique repose aussi sur le transfert NOE entre le ligand et la protéine. Un filtre T_2, de type CPMG (Carr–Purcell–Meiboom–Gill), préalable au temps de mélange τ_m, pendant lequel s'exprime la relaxation croisée inter-moléculaire entre ligand et récepteur, permet d'atténuer le signal de la protéine. La soustraction avec un spectre de référence où le temps de mélange est inversé avec le filtre T_2 permet de détecter uniquement les molécules en interaction avec le récepteur. Un gradient de "purge" peut être appliqué pendant le temps de mélange afin de défocaliser les composantes dans le plan (x, y) (figure 5.9 page suivante).

La RNP ne permet pas de réaliser un *epitope mapping* des zones d'interaction du ligand, du fait notamment du phénomène de diffusion de spin [104].



(a) *Epitope mapping* STD (en noir) et WaterLOGSY (en gris) du naproxène en interaction avec la HSA



(b) Formule développée plane du naproxène

FIGURE 5.8 – Comparaison de l'*epitope mapping* obtenu par STD (en noir) avec celui obtenu par la WaterLOGSY (en gris) sur le modèle HSA·naproxène. [HSA] = $30\mu M$; R = [naproxène]/[HSA] = 30

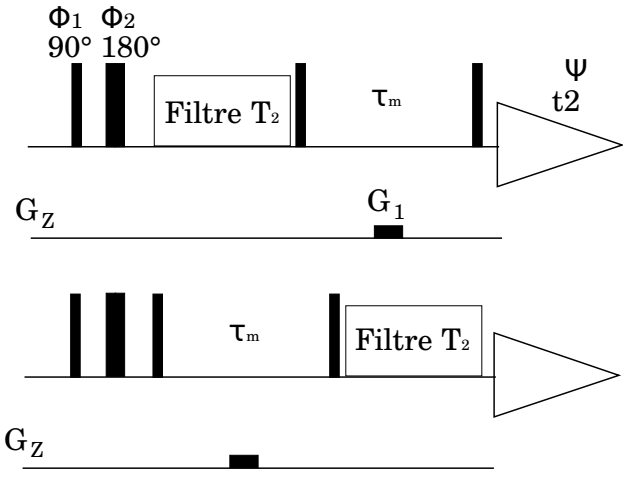


FIGURE 5.9 – Séquence RMN d'impulsions de l'expérience 1D RNP ^1H.

5.4 Utilisation des données RMN comme source d'informations structurales

Nous allons ici nous focaliser sur l'utilisation des données issues des spectres STD et WaterLOGSY afin de déterminer des informations sur la structure du ligand en interaction avec la protéine. L'*epitope mapping* issu des données de STD ou WaterLOGSY permet de déterminer les protons du ligand les plus en interaction avec la protéine (voir figure 5.8 page précédente). La dépendance des effets mis en jeu dans l'expérience avec la distance inter-protons entre ligand et protéine est en $1/r^6$ (voir équation 5.9 page 98).

La méthode CORCEMA [80] permet de calculer l'intensité du signal STD à partir des coordonnées atomiques du complexe. L'utilisation des matrices de relaxation permet aussi d'affiner la conformation du ligand lié en utilisant les intensités STD expérimentales [81].

Les calculs de *docking* génèrent un modèle du complexe entre le ligand étudié et son récepteur. La combinaison des méthodes de *docking* avec les données expérimentales RMN permettent de valider et d'affiner le modèle ainsi obtenu. La mesure des distances minimales inter-protons ligand·protéine r suivi du calcul de $1/r^6$ permet d'obtenir simplement un *epitope mapping* théorique, comparable avec les données expérimentales.

Les algorithmes de *docking* (voir partie 3.1 page 41) génère un ensemble de solution pour un complexe donné. La sélection de la meilleure solution n'est pas un problème trivial et est souvent lié au problème du calcul des énergies d'interaction. L'utilisation des données d'*epitope mapping* expérimentales peut permettre de sélectionner les poses de *docking* les plus proches de celles-ci.

Cependant la comparaison des données théoriques et expérimentales peut être rendu difficile pour plusieurs raisons :

- la détermination de l'*epitope mapping* expérimental peut être rendue difficile du fait de déformations de la ligne de base
- les poses de *docking* générés peuvent être erronées, au moins sur une partie du ligand, du fait des nombreuses approximations réalisées lors des calculs.

Afin d'éviter ces problèmes, une bonne élimination du solvant est nécessaire lors de l'acquisition des spectres et le nombre de poses générées lors du *docking* doit être élevé (par exemple 200 poses) afin de bien échantillonner l'espace conformationnel du ligand au sein du récepteur. L'augmentation du nombre de poses générées entraîne une augmentation du nombre de données à comparer. Le classement des poses par rapport aux données expérimentales peut être réalisé par l'utilisation des cartes de Kohonen (voir paragraphe 4.2.2

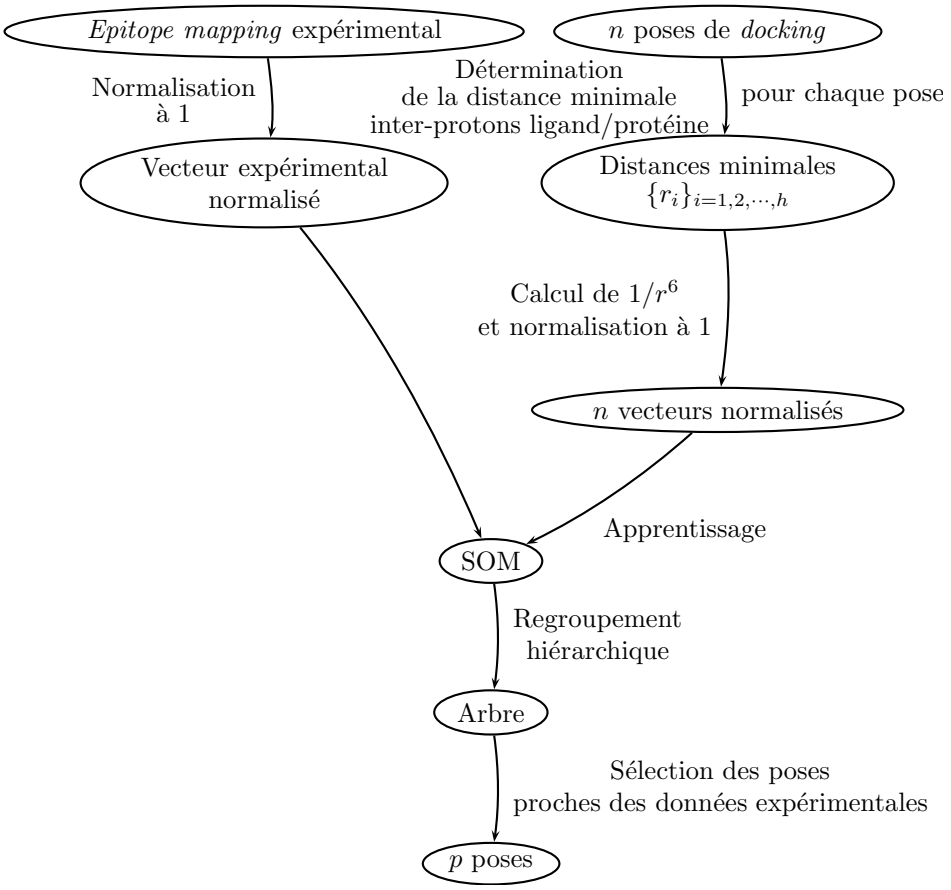


FIGURE 5.10 – Méthode de sélection des poses de *docking* en accord avec les données d'*epitope mapping* expérimentales

page 74). Celles ci permettent de regrouper des éléments semblables malgré les imperfections générées par les conditions expérimentales ou des calculs trop approximatifs. La figure 5.10 donne la procédure de sélection des poses de *docking* selon les données d'*epitope mapping*.

5.5 Conditions expérimentales

5.5.1 Équipements utilisés

Les expérience de RMN ont été réalisées à 500,13 MHz pour le proton ^1H sur un spectromètre RMN Bruker Avance 500 (Bruker Biospin, Wissembourg, France) équipé d'une sonde TXI ^1H, ^{13}C, ^{15}N et d'une station de

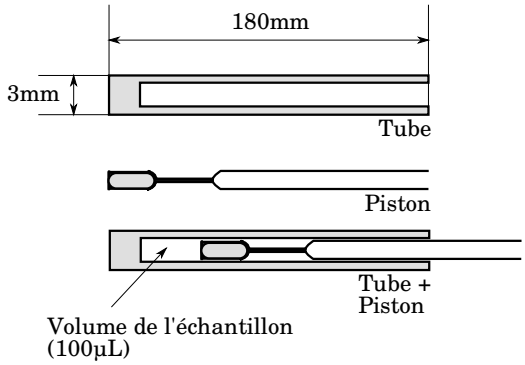


FIGURE 5.11 – Vue en coupe d'un tube Shigemi de 3mm de diamètre extérieur.

travail Linux. Des tubes shigemi (Shigemi, Inc., Allison Park, PA, USA – figure 5.11) d'un diamètre extérieur de 3mm ont été utilisé afin de limiter la consommation de protéine et de ligand. Ce type de tube permet de diminuer le volume de l'échantillon de $500\mu L$ à $100\mu L$, et possède une susceptibilité magnétique identique à celle du solvant, en l'occurrence $H_2O/^2H_2O$ (90/10), limitant ainsi les effets de bord. Les spectres ont été traités par le logiciel XWIN-NMR 3.5 (Bruker Biospin, Wissembourg, France) et analysés par le logiciel NMRnotebook 2.5 (NMRTEC, Illkirch-Graffenstaden, France).

5.5.2 Préparation des échantillons

Le complexe intégrase (IN), LEDGF a été préparé comme décrit dans l'article de Michel *et al.* [123]. Les *tags* GST [2] de l'intégrase et His_6 du LEDGF n'ont pas été retirés ce qui permet d'augmenter la masse du récepteur, conditions favorables aux expériences de *transfer-NMR*. Le complexe IN_2·$LEDGF_2$ obtenu possède une masse molaire de 250kDa. Le complexe fourni par la société CellVir SAS est en solution dans du tampon HEPES [3] (50mM HEPES pH7,0 ; 150mM NaCl ; 5mM $MgCl_2$; 2mM β-mercaptoethanol). Cette solution a ensuite été dialysée contre du tampon phosphate (50mM phosphate pH7,3 ; 150mM NaCl). La dialyse a été réalisée dans des boutons de dialyse de 500μL avec une membrane de dialyse de *cutoff* de 30kDa pendant 2 heures contre 500mL de tampon phosphate. Le volume de la solution de complexe obtenu ayant doublé, la solution est passée en microcon de 500μL de *cutoff* de 3kDa afin de descendre le volume à 200μL par centrifugation (10 000rpm

2. *Glutathione S-Transferase*
3. acide 4-(2-hydroxyéthyl)-1-pipérazine éthane sulfonique

pendant 30 minutes). La concentration protéique de la solution obtenue est de 16μM.

Les différents ligands fournis par la société CellVir SAS ont été dilué préalablement dans du DMSO [4] afin d'obtenir une concentration de 20mM. Des concentrations variables de ligands ont été ajouté dans le tube RMN shigemi contenant 100μL de solution protéique et 10μL de ^2H_2O afin d'obtenir des rapports finals $[L]/[P]$ compris entre 50 et 200.

4. diméthylsulfoxyde

Troisième partie

Résultats

Chapitre 6

Validations et applications du logiciel AuPosSOM

6.1 Criblage de la chimiothèque du laboratoire

Le logiciel AuPosSOM a initialement était testé sur la chimiothèque du laboratoire (voir paragraphe 3.2.1 page 56). Les trois cibles décrites dans le paragraphe 3.2.1 page 56 ont été utilisées comme récepteur pour les calculs de *docking*, à savoir, la protéase du VIH-1 (code pdb : 2bpw), la transcriptase inverse du VIH-1 (1uwb) et la thrombine humaine (1afe) (voir tableau 3.1 page 65).

Le tracé des courbes de ROC est classiquement utilisé lorsque chaque élément à classer est caractérisé par un score (paragraphe 3.2.2 page 58). Cependant il est par conséquent moins aisé de calculer les coordonnées des points de la courbe de ROC lors du classement des molécules selon les contacts. L'approche décrite dans la publication du logiciel AuPosSOM dans le journal *Bioinformatics* [14] (voir page 129) utilise une succession d'apprentissage avec des sous ensembles de la base de données criblée. Après chaque apprentissage l'ensemble des vecteurs d'entrée, incluant toute la chimiothèque, est calibré sur la carte résultant de l'apprentissage avec le sous ensemble. L'ensemble de plus faible γ (voir équation 3.27 page 62) est sélectionné et la sensibilité Se et la spécificité Sp est reporté dans l'espace de ROC ($Se = f(1 - Sp)$). Cette procédure est résumée par le diagramme présenté en figure 6.1 page suivante.

Les résultats obtenus pour la protéase du VIH-1 sont présentés en figure 6.2 page 115. Les courbes de ROC obtenus avec l'algorithme de *docking* implémenté dans le logiciel AutoDock (figure 6.2 page 115 A et B)

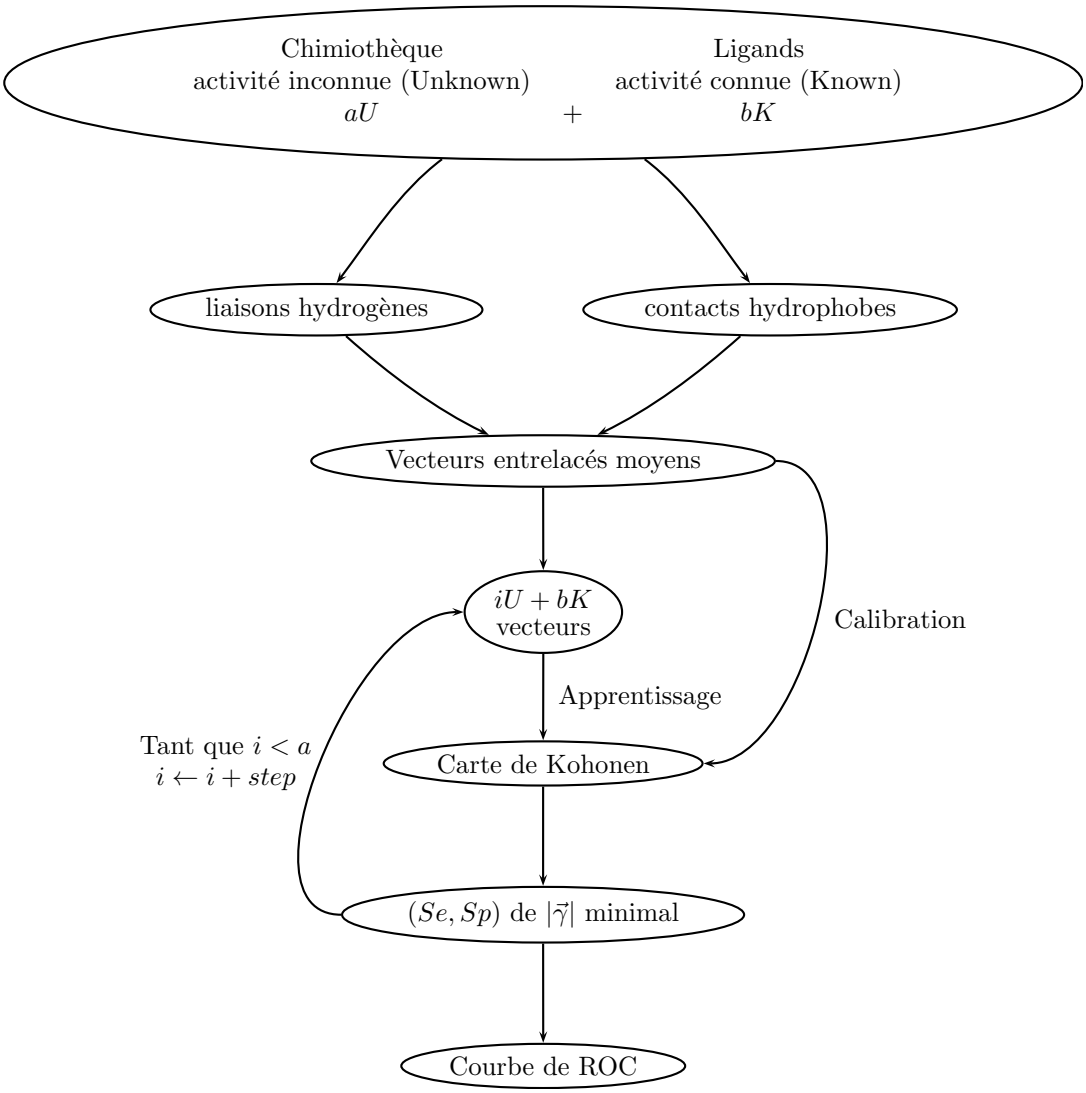


FIGURE 6.1 – Traçage itératif des courbes de ROC en fonction de l'ensemble utilisé pour l'apprentissage.

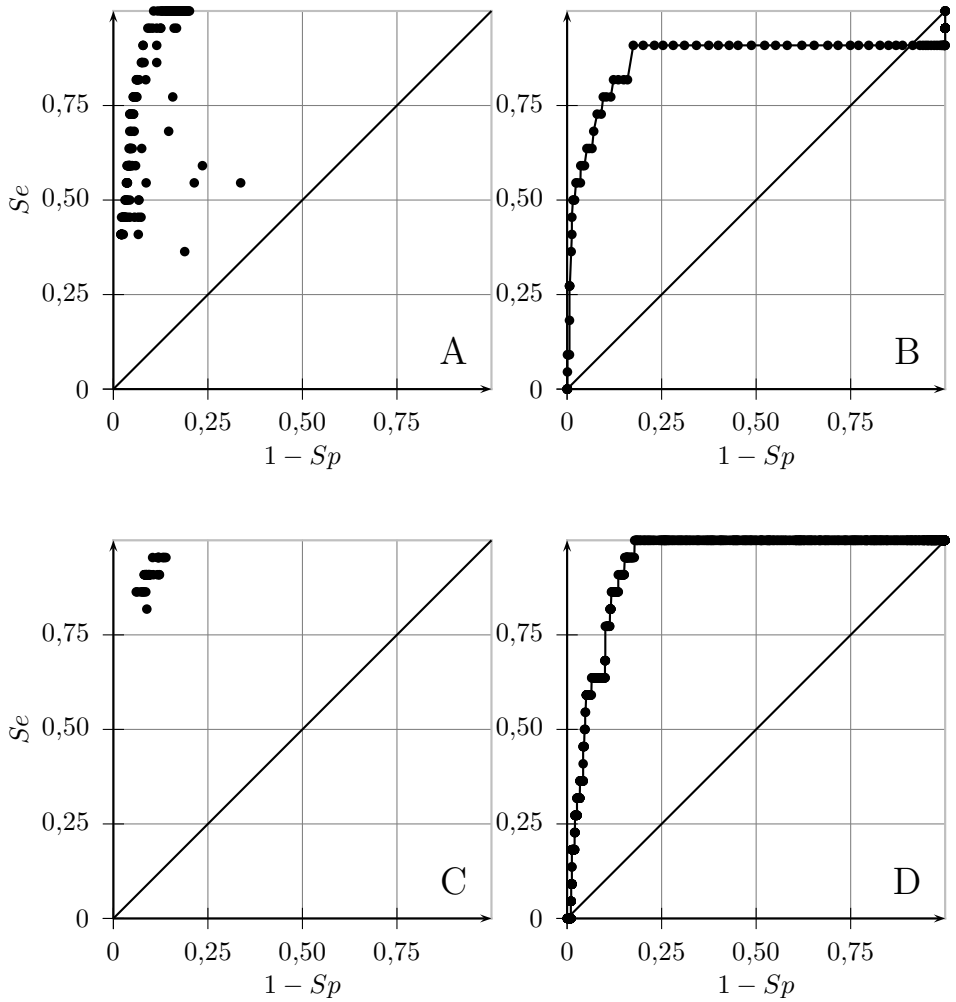


FIGURE 6.2 – Courbes de ROC obtenues à partir de poses de *docking* générées par AutoDock 4.0 (**A** et **B**) et surflex-dock 2.4 (**C** et **D**) sur la protéase du VIH-1. **A** : Courbe de ROC obtenue à partir du logiciel AuPosSOM sur 20 poses générées par AutoDock. **B** : Courbe de ROC obtenue en utilisant la fonction de score implémentée dans AutoDock. **C** : Courbe de ROC obtenue à partir du logiciel AuPosSOM sur 20 poses générées par Surflex-dock. **D** : Courbe de ROC obtenue en utilisant la fonction de score implémentée dans Surflex-dock.

montrent une nette amélioration de la détection des molécules actives lorsque le protocole mis en oeuvre par le logiciel AuPosSOM est utilisé. En effet le tri des molécules par la fonction de score du logiciel AutoDock ne permet pas de trouver l'ensemble des molécules actives ($Se = 0,91$ et $1 - Sp = 0,18$) (figure 6.2 page précédente B). Ceci est dû à une mauvaise estimation de l'énergie d'interaction pour deux composés. Au contraire l'analyse des contacts permet de regrouper l'ensemble des molécules actives ($Se = 1$) avec une faible proportion de faux positifs ($1 - Sp = 0,10$) (figure 6.2 page précédente A). Par conséquent les poses générées par AutoDock pour les deux molécules non sélectionnées par le filtre de score possèdent des contacts avec la protéine qui sont similaires aux contacts effectués par les molécules actives. Cependant le criblage effectué par le filtre de score reste correcte et comparable à des résultats publiés auparavant sur cette même cible [20].

La courbe de ROC caractéristique du crible effectué par le logiciel AuPosSOM sur la protéase (figure 6.2 page précédente A) montre une dispersion importante des points dans l'espace de ROC. Cette dispersion est essentiellement créée lors des étapes d'apprentissage avec peu de vecteurs d'entrée ($< 25\%$ de l'ensemble des vecteurs d'entrée). Lorsque le nombre de vecteurs d'entrée utilisés pour l'apprentissage est faible, l'effet de l'ajout, à cet ensemble, de nouveaux vecteurs aléatoirement sélectionnés entraîne de nombreuse modifications sur le résultat du tri. Ceci se reflète par une dispersion des points dans l'espace de ROC. Lorsque l'ensemble des vecteurs d'entrée est utilisé pour l'apprentissage – comportement par défaut du logiciel – le point de sensibilité $Se = 1$ et de spécificité $Sp = 0,9$ est obtenu.

Le point caractérisé par un $|\vec{\gamma}|$ (équation 3.27 page 62) minimal n'est pas nécessairement le point de plus forte sensibilité (voir tableau 6.1 page suivante). Cependant ce coefficient permet de comparer facilement deux protocoles de criblage virtuel, l'aire sous la courbe n'étant pas calculable pour les courbes de ROC obtenues avec l'analyse des contacts. Le coefficient γ est deux fois moins grand pour le regroupement par contact par rapport au score AutoDock ce qui reflète une augmentation à la fois de la sensibilité et de la spécificité du filtre par contact.

Le regroupement obtenu par l'analyse des contacts intermoléculaires est représenté sur la figure 6.3 page 118. Il résulte de l'entraînement d'une carte de Kohonen de dimension 5×4 avec l'ensemble des vecteurs correspondant à l'analyse de l'ensemble des contacts ligands/récepteur. Les paramètres de l'apprentissage (en deux phase) permettant l'obtention de l'arbre présenté sont les suivants :

- Phase 1 :
  - 1 000 itérations
  - Décroissance exponentielle du coefficient d'apprentissage α de 0,2 à

Modèle	$card(\mathbb{A})$	SOM *clustering*			*k-means clustering*			Filtre de score AutoDock		
		Se	Sp	γ	Se	Sp	γ	Se	Sp	γ
Protéase VIH-1 (2bpw)	22	0,95	0,91	0,10	0,82	0,94	0,19	0,91	0,82	0,20
Transcriptase inverse VIH-1 (1uwb)	14	0,57	0,90	0,44	0,71	0,70	0,42	0,50	0,63	0,62
Thrombine humaine (1afe)	20	0,85	0,85	0,21	0,80	0,90	0,22	0,80	0,85	0,25

TABLE 6.1 – Sensibilité Se (équation 3.25 page 60) et spécificité Sp (équation 3.26 page 60) correspondant au point de $|\vec{\gamma}|$ (équation 3.27 page 62) le plus faible. $card(\mathbb{A})$ est le nombre de molécules actives introduites dans la chimiothèque (figure 3.9 page 60).

0
  - Décroissance exponentielle du rayon σ de 6 à 1
- Phase 2 :
  - 10 000 itérations
  - Décroissance exponentielle du coefficient d'apprentissage α de 0,02 à 0
  - Décroissance exponentielle du rayon σ de 3 à 1

Ces paramètres sélectionnés par défaut dans le logiciel AuPosSOM permettent d'obtenir une bonne convergence de la carte de Kohonen. Ainsi diverses initiations aléatoires de la carte mènent à une topologie identique de l'arbre obtenu. Le choix d'une carte de taille réduite (5×4) menant à l'obtention de vingt groupes au maximum est aussi à l'origine d'une bonne convergence du calcul. L'analyse de l'arbre obtenu (figure 6.3 page suivante) permet de suite d'isoler un ensemble de molécules ($\mathbb{S}$) contenant l'ensemble des molécules actives ($\mathbb{A} \subset \mathbb{S}$). Cet ensemble est mis en évidence par la couleur magenta. Les branches en couleur cyan indiquent les molécules actives $\mathbb{A}$, les autres branches de cet ensemble (en magenta) sont des molécules d'activité inconnue, potentiellement actives. Dans cet arbre, la longueur des branches est proportionnelle à la divergence des modes d'interaction des différents groupes (20 groupes au total). L'ensemble $\mathbb{S}$ est éloigné du reste de l'arbre contenant aucune molécule active connue ($\mathbb{D} \backslash \mathbb{S}$). Ainsi l'ensemble $\mathbb{S}$ se distingue du reste de l'arbre par des empreintes d'interactions très différentes. Il est possible de réaliser une empreinte consensus qui correspond à la moyenne des vecteurs de l'ensemble $\mathbb{S}$ caractérisant cet ensemble. Si on reporte cette empreinte consensus sur la structure cristallographique de la protéase du VIH-1 nous obtenons

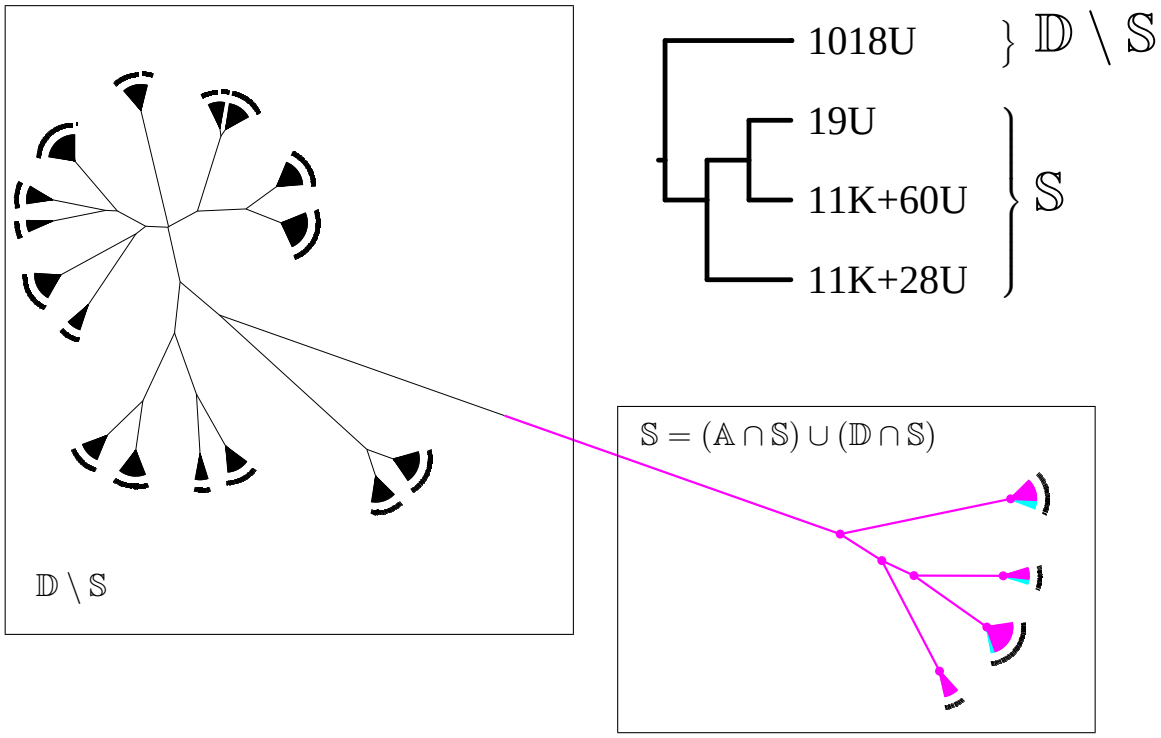


FIGURE 6.3 – Représentation, sous forme de graphique en arbre non enraciné, du classement des molécules dockées par le logiciel AutoDock sur la protéase du VIH-1. Chaque feuille est composé d'un grand nombre de molécule. La longueur des branches est proportionnelle à la divergence des contacts effectués par les molécules. Les ensembles notés sont ceux définis sur la figure 3.9 page 60. Le dendrogramme représente une vue simplifiée de l'arbre non enraciné et donne le nombre d'élément dans chaque groupe.

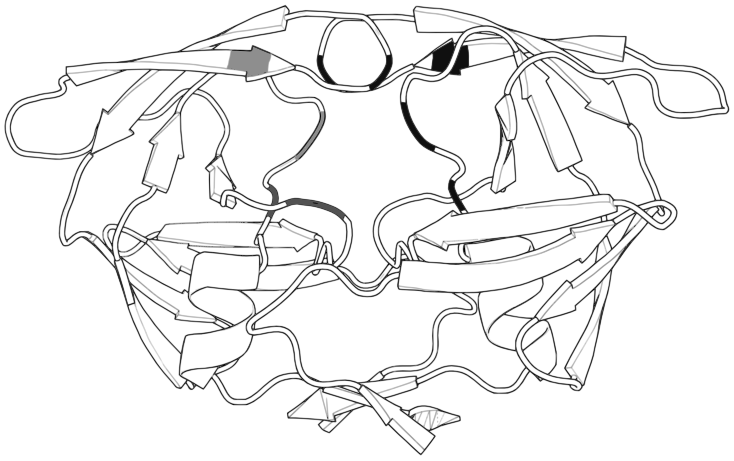


FIGURE 6.4 – Structure cristallographique (code PDB : 2bpw) de la protéase du VIH-1. Sur cette structure dimérique est reportée sur le monomère de gauche l'empreinte consensus représentative de l'ensemble $\mathbb{S}$ de la figure 6.3 page ci-contre
et sur le monomère de droite l'ensemble des mutations entraînant une résistance aux anti-protéases. Les acides aminés concernés sont : I47, I50, V82 et I84 [83]. L'alanine 28 est détectée par l'analyse de contacts mais n'est pas associée à des résistances.

la figure 6.4. L'empreinte consensus, représentée sur le monomère de gauche, montre de fortes similitudes avec les mutations de la protéase associées à une résistance aux anti-protéases. Cependant une différence majeure est à remarquer au niveau de l'alanine 28. Cet acide aminé est contacté par les molécules de l'ensemble $\mathbb{S}$ mais sa mutation n'est associée à aucune résistance aux anti-protéases. Des données bibliographiques montrent que cet acide aminé est essentiel à la structuration de cette protéine et que sa mutation entraîne la dénaturation complète de celle-ci. Cet acide-aminé est d'ailleurs visé afin de construire de nouvelles molécules échappant à la résistance aux anti-protéases [178].

Afin de ne pas limiter l'étude à un seul logiciel de *docking*, le programme surflex-dock a été utilisé pour docker la même chimiothèque sur la protéase du VIH-1. Les courbes de ROC obtenues lors de l'analyse de contacts ou lors du filtre de score avec la fonction de score Surflex-Dock sont représentées sur les figures 6.2 page 115 C et D respectivement. La fonction de score Surflex-Dock permet de retrouver l'ensemble des molécules actives ($Se = 1$) par opposition à la fonction de score implémentée dans AutoDock (figure 6.2 page 115 B). Du fait que les poses générées par Surflex-Dock sont différentes

de celles générées par AutoDock nous pouvons réaliser une nouvelle analyse de contacts avec le logiciel AuPosSOM (figure 6.2 page 115 C). Cette analyse donne un regroupement plus spécifique à sensibilité équivalente que le tri par le score. La comparaison de l'analyse de contact effectuée après un *docking* par Surflex-Dock avec celle effectuée après un *docking* avec AutoDock (figure 6.2 page 115 A et C) montre une dispersion moindre dans l'espace de ROC pour Surflex-Dock. Ceci montre une convergence plus grande sur les vingts poses de *docking* générées par Surflex-Dock. Cependant une molécule ne peut pas être sélectionné par l'analyse des poses ce qui montre un problème de positionnement pour celle ci.

Hormis les cartes de Kohonen, il existe de nombreuses autres méthodes de *clustering*. Parmi celles-ci la méthode *k-means* est très utilisée. Nous avons comparé l'efficacité de cette méthode par rapport aux cartes de Kohonen. Afin d'étudier l'effet des paramètres k – le nombre de groupes dans la méthode des k-moyennes (*k-means*) – et (X, Y) – la taille de la carte de Kohonen – les courbes de ROC de la figure 6.5 page suivante sont données pour différentes valeurs de ces paramètres. Les distributions en abscisse et en ordonnée des points de l'espace de ROC sont aussi données. Les cartes de Kohonen donnent clairement de meilleures résultats ($Se = 1$) que la méthode *k-means* ($Se = 0.82$). De plus la dispersion des résultats est moins importante pour les cartes de Kohonen. De plus les résultats les plus fréquents correspondent au point de sensibilité et de spécificité les plus élevées. Les points les plus représentés sont caractérisés par les valeurs $Se = 1\,;\ Sp = 0,87$ et $Se = 0,73\,;\ Sp = 0,95$ pour les méthodes SOM et *k-means* respectivement. Le coefficient γ est meilleur pour le SOM ($\gamma = 0,10$) par rapport à la méthode des k-moyennes ($\gamma = 0,19$) (voir tableau 6.1 page 117).

Les courbes de ROC obtenues pour le *docking* avec AutoDock de la chimiothèque sur la transcriptase-inverse du VIH-1 sont présentées en figure 6.6 page 122. La courbe de ROC obtenue par filtre de score avec la fonction de score implémentée dans AutoDock (figure 6.6 page 122 B) montre clairement la faiblesse de cette fonction dans la discrimination des composés actifs. Cependant une nette amélioration est observée avec le tri par analyse de contacts (figure 6.6 page 122 A). Le coefficient γ passe de 0,62 à 0,44 si l'analyse de contact est utilisée, relativement au score (voir tableau 6.1 page 117).

La comparaison de la méthode *k-means* avec la méthodes des cartes de Kohonen ne montre pas ici une nette amélioration lors de l'utilisation de cette dernière. Le coefficient γ est sensiblement le même : 0,42 pour la méthode *k-means* contre 0,44 pour les cartes de Kohonen (voir tableau 6.1 page 117). Les représentations de ROC obtenues avec ces deux méthodes pour la transcriptase inverse sont présentées en figure 6.7 page 123. Nous constatons encore

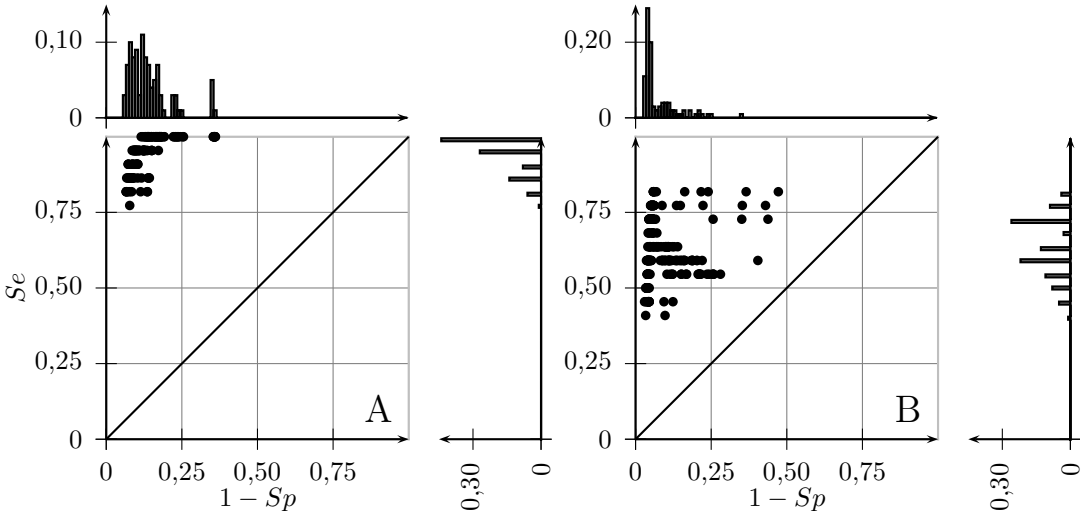


FIGURE 6.5 – Comparaison de la méthode SOM *clustering* (**A**) et *k-means clustering* (**B**) pour le *docking* par AutoDock de la chimiothèque sur la protéase du VIH-1. Les courbes de ROC sont données pour différentes tailles de cartes de Kohonen ($(X, Y) = \{(i, j) \in \mathbb{N}^2; 2 \leq i \leq j \leq 6\}$) et différentes valeurs de k ($k = \{2, 3, \cdots, 30\}$). Pour chaque valeur de paramètre le calcul est répété dix fois. Pour chaque graphique la distribution des points en abscisse et en ordonnée est donnée.

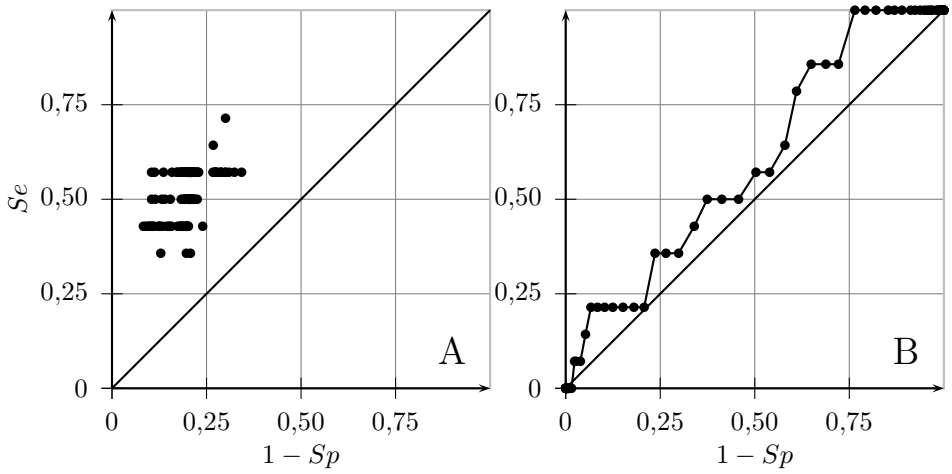


FIGURE 6.6 – Courbes de ROC obtenues à partir de poses de *docking* générées par AutoDock 4.0 (**A** et **B**) sur la transcriptase inverse du VIH-1. **A** : Courbe de ROC obtenue à partir du logiciel AuPosSOM sur 20 poses générées par AutoDock. **B** : Courbe de ROC obtenue en utilisant la fonction de score implémentée dans AutoDock.

une plus grande dispersion des points dans l'espace de ROC pour la méthode des k-moyennes. La méthode SOM montre aussi une distribution des points plus proche de la zone de haute spécificité ($Sp > 0,75$).

Le logiciel a aussi été testé sur la thrombine humaine. Les courbes de ROC obtenues sont présentées en figure 6.8 page 124. Sur cette cible l'analyse par score donne des résultats satisfaisants. L'analyse de contacts par SOM *clustering* améliore légèrement la sensibilité du criblage à spécificité égale : $Se = 0,85$ pour la méthode SOM contre $Se = 0,80$ pour l'analyse par score pour une spécificité $Sp = 0,85$ (voir tableau 6.1 page 117).

Comme pour la transcriptase inverse l'utilisation du SOM ne montre pas d'amélioration sensible par rapport à la méthode *k-means* : le coefficient γ du SOM (0,21) est quasiment égale à celui de la méthode *k-means* (0,22) (voir tableau 6.1 page 117). Cependant la méthode *k-means* montre encore une plus grande dispersion des points dans l'espace de ROC (figure 6.9 page 125) suite aux variations des paramètres de la méthode et des initialisations aléatoires. L'utilisation du SOM pour cette cible montre l'apparition de deux ensembles de points dans l'espace de ROC. Un ensemble est proche de la diagonale tandis que l'autre s'approche de la région de plus forte spécificité. La région proche de la diagonale correspond à des cartes de faibles dimensions. Un apprentissage avec les paramètres par défaut (carte de dimension 5×4) donne

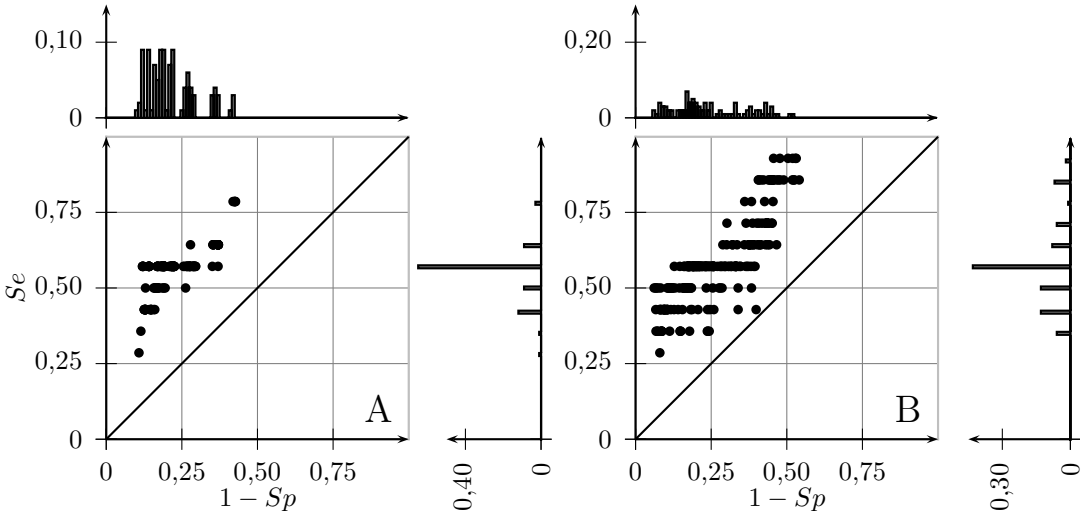


FIGURE 6.7 – Comparaison de la méthode SOM *clustering* (**A**) et *k-means clustering* (**B**) pour le *docking* par AutoDock de la chimiothèque sur la transcriptase inverse du VIH-1. Les courbes de ROC sont données pour différentes tailles de cartes de Kohonen ($(X, Y) = \{(i, j) \in \mathbb{N}^2; 2 \leq i \leq j \leq 6\}$) et différentes valeurs de k ($k = \{2, 3, \cdots, 30\}$). Pour chaque valeur de paramètre le calcul est répété dix fois. Pour chaque graphique la distribution des points en abscisse et en ordonnée est donnée.

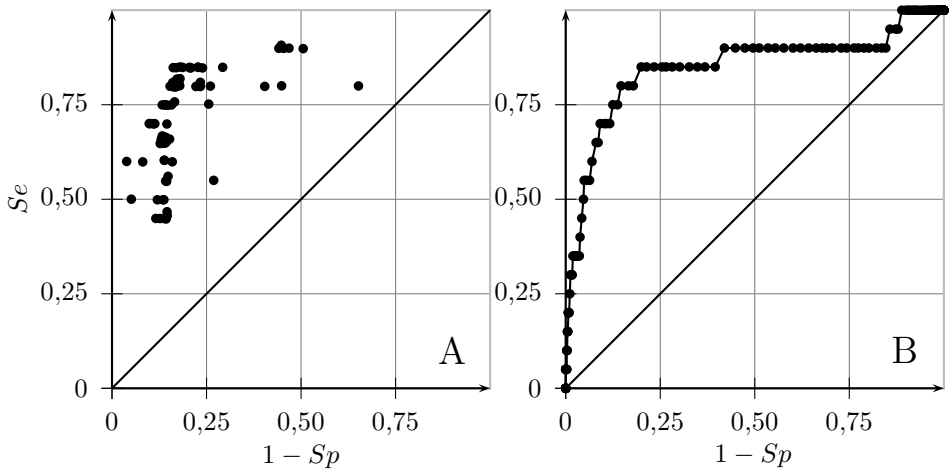


FIGURE 6.8 – Courbes de ROC obtenues à partir de poses de *docking* générées par AutoDock 4.0 (**A** et **B**) sur la thrombine humaine. **A** : Courbe de ROC obtenue à partir du logiciel AuPosSOM sur 20 poses générées par AutoDock. **B** : Courbe de ROC obtenue en utilisant la fonction de score implémentée dans AutoDock.

le point de sensibilité $Se = 0,85$ et de spécificité $Sp = 0,85$. Ainsi la taille de la carte ne doit pas être trop réduite afin d'obtenir des tris sensibles et spécifiques.

Pour l'ensemble des modèles testés ici, l'analyse AuPosSOM est à l'origine d'un tri plus sensible et plus spécifique que l'analyse par score utilisant la fonction implémentée dans AutoDock 4.0. De même l'utilisation du logiciel Surflex-Dock, bien que donnant des résultats plus sensibles et spécifiques que AutoDock ne permet pas, pour la protéase du VIH-1, de dépasser l'analyse par contact.

La comparaison de l'analyse des contacts par SOM avec celle effectuée par la méthode *k-means* montre la supériorité du SOM pour le modèle de la protéase. Cependant pour le modèle de la transcriptase inverse et de la thrombine humaine la supériorité du SOM n'est pas sensible mais donne des résultats comparables à la méthode *k-means*. Du fait que les seuls variations entre les modèles sont les vecteurs d'entrée l'analyse précise de ceux ci est nécessaire afin de comprendre ces observations.

La dispersion au sein des vecteurs et entre les vecteurs est un paramètre important qui peut influencer la qualité du regroupement effectué par les cartes de Kohonen ou la méthode *k-means*. La dispersion des données au sein d'un vecteur peut être quantifiée par le calcul de l'entropie de Shannon

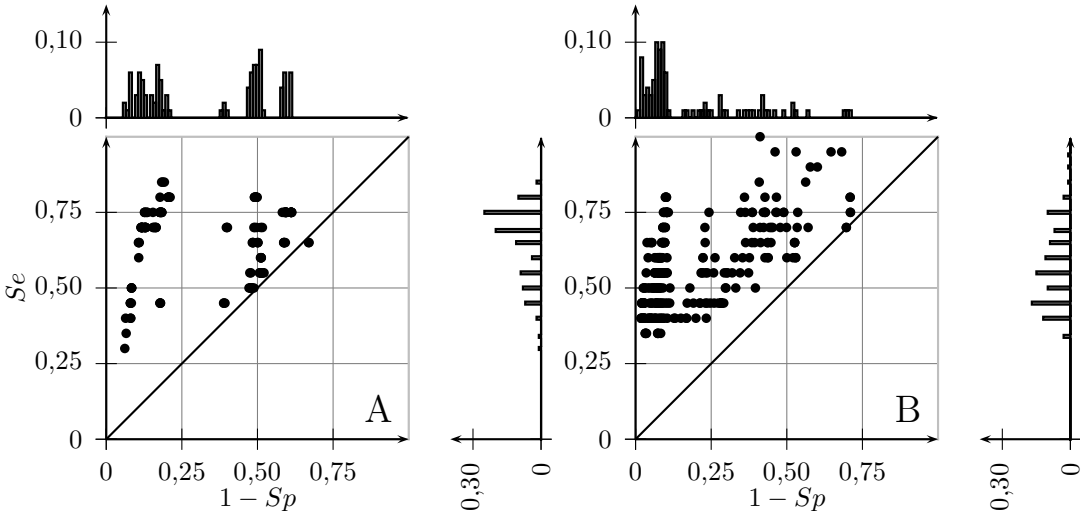


FIGURE 6.9 – Comparaison de la méthode SOM *clustering* (**A**) et *k-means clustering* (**B**) pour le *docking* par AutoDock de la chimiothèque sur la thrombine humaine. Les courbes de ROC sont données pour différentes tailles de cartes de Kohonen ($(X, Y) = \{(i, j) \in \mathbb{N}^2; 2 \leq i \leq j \leq 6\}$) et différentes valeurs de k ($k = \{2, 3, \cdots, 30\}$). Pour chaque valeur de paramètre le calcul est répété dix fois. Pour chaque graphique la distribution des points en abscisse et en ordonnée est donnée.

[158]. Elle est calculée par la relation :

$$SE = -\sum_{i=1}^{N} p_i \log(p_i) \tag{6.1}$$

avec p_i la probabilité de rencontrer, dans le vecteur considéré, une valeur comprise dans l'intervalle i ($i = \{1, 2, \cdots, N\}$). Si la distribution des valeurs du vecteur considéré est représentée sous forme d'histogramme alors p_i est donnée par la relation :

$$p_i = \frac{c_i}{\sum_{i=1}^{N} c_i} \tag{6.2}$$

où c_i est la valeur donnée par l'histogramme pour l'intervalle i. L'entropie de Shannon peut être normalisée à 1 grâce au facteur ρ définit ainsi :

$$\rho = \frac{SE_{rand} - SE}{SE_{rand}} \tag{6.3}$$

avec SE_{rand} l'entropie de Shannon d'un vecteur dont les valeurs des éléments sont distribuées aléatoirement (*random*) :

$$SE_{rand} = -\log\left(\frac{1}{N}\right) \tag{6.4}$$

Ainsi,

$$\rho = 1 - \frac{SE}{\log(N)}; \rho \in [0; 1] \tag{6.5}$$

L'entropie de Shannon est ainsi normalisée par rapport à une distribution aléatoire des données. $\rho = 0$ pour une distribution aléatoire $\rho = 1$ si le vecteur comporte que des éléments identiques.

La distribution de ρ, notée $p(\rho)$ peut être approximée par une loi normale :

$$p(\rho) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\rho - \bar{\rho})^2}{2\sigma^2}\right) \tag{6.6}$$

avec $\bar{\rho}$ la valeur moyenne de ρ et σ l'erreur standard sur ρ.

La figure 6.10 page suivante montre la distribution du facteur σ pour les trois modèles considérés. Dans cette figure ρ indique l'ordre des données au sein des vecteurs (dispersion intra-vectorielle). Plus ρ est grand moins il y a de valeurs différentes au sein du vecteur considéré. σ indique la dispersion des vecteurs entre eux (dispersion inter-vectorielle). Les paramètres $\bar{\rho}$ et σ pour la transcriptase inverse et la thrombine sont sensiblement les mêmes. La protéase se distingue de ces modèles par une dispersion intra-vectorielle et inter-vectorielle plus importantes par rapport aux deux autres modèles.

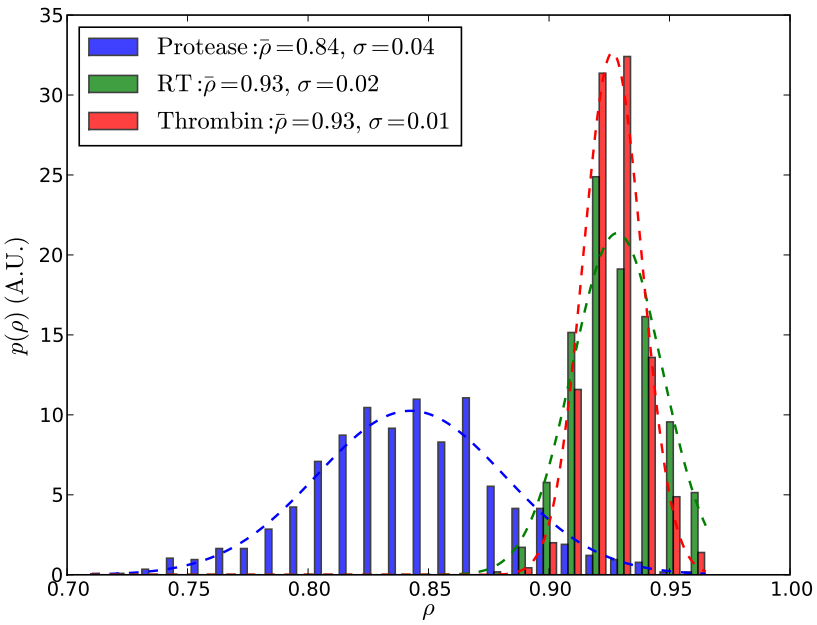


FIGURE 6.10 – Dispersion des vecteurs d'entrée pour la protéase (*protease*), la transcriptase inverse (*RT*) et la thrombine (*thrombin*). Les paramètres des lois normales correspondantes ($\bar{\rho}$ et σ) sont donnés dans l'encart.

L'avantage de l'analyse par les cartes de Kohonen semble être mis en avant par comparaison à la méthode *k-means* lorsque la dispersion des vecteurs d'entrée augmente. C'est ainsi que le *clustering* obtenu avec le modèle de la protéase est meilleur que lorsque la méthode *k-means* est utilisée. L'utilisation de la méthode *kmeans* semble du même niveau d'efficacité que le SOM lorsque les vecteurs d'entrée sont peu dispersés donc les poses de *docking* similaires.

L'ensemble de ces résultats a donné lieu à une publication [14] présentée dans les pages suivantes.

G. Bouvier, N. Evrard-Todeschi, J.-P. Girault, and G. Bertho. Automatic clustering of docking poses in virtual screening process using self-organizing map. *Bioinformatics*, 26(1) :53–60, 2010.

BIOINFORMATICS ORIGINAL PAPER Vol. 26 no. 1 2010, pages 53–60
doi:10.1093/bioinformatics/btp623

Structural bioinformatics

Automatic clustering of docking poses in virtual screening process using self-organizing map

Guillaume Bouvier, Nathalie Evrard-Todeschi, Jean-Pierre Girault and Gildas Bertho*

Laboratoire de Chimie et de Biochimie Pharmacologiques et Toxicologiques, Unité Mixte de Recherche 8601, Centre National de la Recherche Scientifique (CNRS), Université Paris Descartes, 45 rue des Saints-Pères, 75006 Paris, France

Received on June 2, 2009; revised on September 22, 2009; accepted on October 25, 2009

Advance Access publication November 12, 2009

Associate Editor: Alfonso Valencia

ABSTRACT

Motivation: Scoring functions provided by the docking software are still a major limiting factor in virtual screening (VS) process to classify compounds. Score analysis of the docking is not able to find out all active compounds. This is due to a bad estimation of the ligand binding energies. Making the assumption that active compounds should have specific contacts with their target to display activity, it would be possible to discriminate active compounds from inactive ones with careful analysis of interatomic contacts between the molecule and the target. However, compounds clustering is very tedious due to the large number of contacts extracted from the different conformations proposed by docking experiments.

Results: Structural analysis of docked structures is processed in three steps: (i) a Kohonen self-organizing map (SOM) training phase using drug–protein contact descriptors followed by (ii) an unsupervised cluster analysis and (iii) a Newick file generation for results visualization as a tree. The docking poses are then analysed and classified quickly and automatically by AuPosSOM (Automatic analysis of Poses using SOM). AuPosSOM can be integrated into strategies for VS currently employed. We demonstrate that it is possible to discriminate active compounds from inactive ones using only mean protein contacts' footprints calculation from the multiple conformations given by the docking software. Chemical structure of the compound and key binding residues information are not necessary to find out active molecules. Thus, contact–activity relationship can be employed as a new VS process.

Availability: AuPosSOM is available at http://www.aupossom.com.

Contact: contact@aupossom.com; gildas.bertho@parisdescartes.fr

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

Virtual screening (VS) is usually described as a cascade of filter approaches to narrow down a set of compounds to be tested for biological activity against the intended drug target. Starting with a fast evaluation of the drug-likeness of compounds, VS is often followed by ligand-based approaches and/or structure-based approaches if the target structure is available (Muegge, 2008).

In structure-based approaches, dockings of molecules onto 3D structure of receptor are made. The improvement of docking algorithms allows to generate ligand conformations similar to crystallographically determined protein/ligand complex structures in most cases. However, scoring functions were less successful at distinguishing the crystallographic conformation from the set of docked poses (Warren *et al.*, 2006). Improvement of scoring functions remains a significant challenge in the application of structure-based VS. However, de-selection of inappropriate compounds is more easily achieved than selection by current scoring schemes. This negative selection is sufficient to reduce the initial large compound database to a shortlist of preferred candidates, but the relative ranking of elements belonging to these shortlist cannot be done using previous criteria. It is a rather common practise to subject a 'reasonably' small number of preselected candidates to visual inspection (Kitchen *et al.*, 2004). This last step is very tedious when the filters used are not restrictive enough. It is dependent on a human interpretation, and user's experience in the field. Such a visual inspection can only be applied to a data sample of perhaps 300–500 compounds (Alvarez and Shoichet, 2005), but it remains very subjective and time consuming. When the number of molecules is very important (more than 500 compounds, where the filters used are not restrictive enough) visual inspection of the complexes is out of question. Accordingly, a reliable filtering has to be performed at the previous steps.

This article demonstrates that it is possible to cluster potentially active molecules using mathematical vectors describing interactions between compounds and targets. Data are clustered using Kohonen self-organizing map (SOM; Kohonen, 2001). This artificial neural network can map high-dimensional data onto a low-dimensional grid such that similar data elements are placed close together. Of particular interest is the method developed by Renner *et al.* (2008). This method is based on a consensus docking scoring using interaction fingerprints comparison and triplet ranking. A SOM analysis of the distribution of docking poses is proposed but not goes any further towards an unsupervised SOM cluster analysis.

The hierarchical clustering of the SOM proposed here bypasses the problem of alternate binding mode identification.

Furthermore, the current article describes the possibility to overcome the problem of scoring function using a statistical analysis of different poses of docking of a same compound onto the target. Iterative learning of the data may neutralize the statistical noise

*To whom correspondence should be addressed.

produced by this analysis. We implement this method in a software called AuPosSOM (Automatic analysis of Poses using SOM). We use for this new software a recent method to perform unsupervised SOM cluster analysis, determine cluster confidence and process result visualization as a tree (Samsonova *et al.*, 2006).

Thus, molecules are clustered according to their binding mode to the macromolecular target. Active compounds and potentially active compounds are then clustered in the same group as their way to explore the macromolecular surface is homologous. AuPosSOM is a flexible tool that can be integrated into strategies for VS currently employed.

2 APPROACH

The goal of VS is to select a small number of potential binders to the receptor of interest from a large source library. Database docking is an approach to solve the problem of identifying compounds in a database of small organic compounds that display favourable interactions (hydrogen bonds, hydrophobic contacts, electrostatic interactions) to the target binding site. A docking program is composed of two elements : an algorithm that explore the conformational space of the small molecule within the binding site and a scoring function that estimate the relative binding energy. This function calculates a crude measure of binding affinity. An ideal scoring function must be able to find out the most favourable docking solutions (called poses) for active molecules in accordance with experimental results (e.g. X-ray structures) and should be able to separate active compounds from inactive ones.

Classical approaches use scoring function in order to discriminate between active compounds and inactive ones. It has been demonstrated that combining multiple scoring functions (consensus scoring) improves the enrichment of true positives (Yang *et al.*, 2005). However, recent validation studies have highlighted the poor performance of currently used scoring functions in estimating binding affinity and hence in ranking large datasets of docked ligands (Waszkowycz, 2008). Energy functions need to be fairly 'soft' so that ligands are not heavily penalized for small errors in the binding geometry. Thus, non-active compounds can obtain a good rank with scoring function but can be excluded using position descriptors.

Another way to classify active molecules and inactive ones is to compare poses between all molecules of the screened database. The process we have developed is shown in Figure 1. The ensemble of active compounds—K compounds (Known)—is defined as $\mathbb{K}$. The ensemble of database compounds with unknown activity—U compounds—is defined as $\mathbb{U}$. For each molecule, an interleaved mean vector is calculated. This vector results from two mean vectors describing hydrogen bonds and hydrophobic contacts involved in protein–molecule interactions. If n docking poses were calculated for each molecule, these vectors describe mean interactions for these n poses. Cardinality of each vector is equal to the number of amino acids in the protein. A subensemble of vectors, composed of all vectors describing known ligands and a random set of i ($i \in [0,a]$, where $a = \mathrm{card}(\mathbb{U})$) vectors from the database, is used for training the SOM. SOM for the Kohonen neural network and TreeSOM algorithm from Samsonova *et al.*, (2006) have been implemented in Python and included in AuPosSOM software. The resulting SOM is then used for clustering all the molecules according to their way to interact with the protein. Mapping of data on a trained SOM is called calibration. Each vector is assigned to the

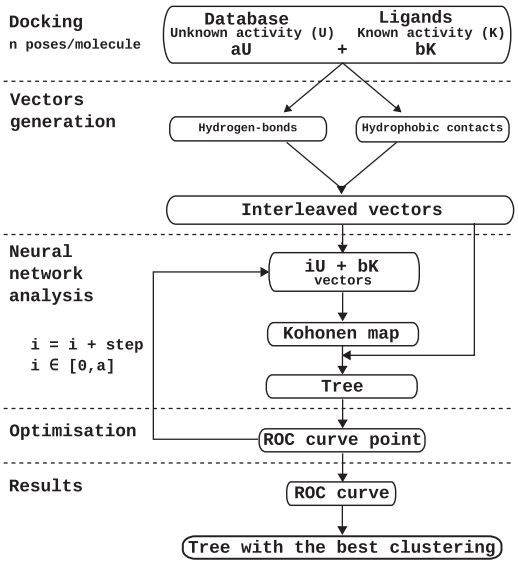


Fig. 1. Diagram illustrating the overall workflow described in this article. From docking results mean vectors describing hydrogen bonds and hydrophobic contacts involved in protein/molecule interactions are computed. Each two vectors are interleaved in order to obtained one vector per molecule in the database (a vector is the mean of the n poses of the docking process). Incremental subensemble are then used to train random SOM. For each trained SOM, all vectors are calibrated on it and then clustered according to the SOM. Each clustering can be visualized as a tree. The best subensemble of vectors to train the SOM is found by plotting a ROC curve.

SOM's node that is most similar to it, so that some nodes may get many data elements, and others none at all. Cluster discovery procedure define clusters as groups of nodes with short distances between them and long distances to the other nodes for a given distance threshold. Python TreeSOM implementation automatically find the best threshold corresponding to the best clustering. For the best clustering, distances between vectors in the same group are minimized and distances to the data in other groups are maximized. AuPosSOM program is able to represent SOM as a tree in Newick file format. At each threshold in the clustering series one or more clusters are split into several subclusters that are represented as a node in the tree. The resultant tree allows a classification of the molecules docked according to their similarity of contacts. Molecules which are clustered with active ones (K molecules) should be probably active too. The best group containing the maximum number of known ligands (K) and minimum number of molecule with unknown activities (U) is found by scanning all the tree, from leaves to root, to calculate for each group sensitivity and specificity. For a given ensemble $\mathbb{E}$, sensitivity (Se) is defined as the number of K compounds which belong to $\mathbb{E}$ divided by the total number of truly active compounds (number of true positive and false negative). If $\mathbb{K}$ is the ensemble of truly active compounds:

$$\mathrm{Se} = \frac{\mathrm{card}(\mathbb{K} \cap \mathbb{E})}{\mathrm{card}(\mathbb{K})} \quad (1)$$

Specificity (Sp) is the number of truly inactive compounds which does not belong to the ensemble $\mathbb{E}$ divided by the total number of molecules with unknown activities. If $\mathbb{U}$ is the ensemble of

molecules with unknown activities:

$$\mathrm{Sp} = \frac{\mathrm{card}(\mathbb{U} \setminus \mathbb{E})}{\mathrm{card}(\mathbb{U})} = 1 - \frac{\mathrm{card}(\mathbb{U} \cap \mathbb{E})}{\mathrm{card}(\mathbb{U})} \quad (2)$$

If Se is plotted against $1-\mathrm{Sp}$ a plot that curves above the diagonal rising from the origin to the upper right corner is obtained. This plot is known as a 'Receiver Operating Characteristic' curve, or ROC curve (Triballeau *et al.*, 2005). On such a graph, a random classification of the compounds would be represented by a diagonal rising from the origin to the upper right corner, whereas a test capable of detecting the correct signal would have an ROC plot that curves above that diagonal. This curve can be used for finding a consensus group with maximum sensitivity and specificity. The best group is the one with the smallest γ, where γ is the norm of the vector between $(0,1)$ point and an ROC curve point :

$$\gamma = \sqrt{(1-\mathrm{Sp})^2 + (1-\mathrm{Se})^2} \quad (3)$$

γ is equal to the Euclidean distance between the characteristics of an ideal test with sensitivity and specificity equal to 1 and the given test studied.

While i value is less than $\mathrm{card}(\mathbb{U})$, i value is incremented by *step* (Fig. 1). The subensemble of vectors obtained is used for training a new random SOM. The resulting SOM is then used for clustering all vectors and the group with minimal γ value is found. The Se and Sp values calculated for the best group for each iteration are then plotted in a new ROC curve (ROC plot in Fig. 1). A γ coefficient can be calculated for each point of the ROC diagram obtained. The best subensemble of vectors used for training the SOM is defined as the subensemble corresponding to the ROC point with minimal γ value.

3 METHODS

3.1 Dataset

AuPosSOM program was tested with three different models: (i) HIV-1 protease, (ii) HIV-1 reverse transcriptase and (iii) human thrombin. For each model, known ligands were used in order to test the ability of AuPosSOM program to discriminate known inhibitors of an enzyme from randomly chosen molecules (decoys). Decoys came from a sub-database of 1108 molecules built from the French National Chemical Library (http://chimiotheque-nationale.enscm.fr). Known ligands with lowest possible pairwise ligand similarity (as measured by pairwise Tanimoto index values) were extracted from the binding database (Liu *et al.*, 2007). For HIV-1 protease, HIV-1 reverse transcriptase and human thrombin, 22, 14 and 20 known ligands were used, respectively. For HIV-1 reverse transcriptase model, non-nucleoside reverse transcriptase inhibitors (NNRTI) were used. For each model ligands lists were built with the respective known ligands and the 1108 decoys.

3.2 Docking procedure

The AutoDock 4.0 package was used for docking simulation (Huey *et al.*, 2007; Morris *et al.*, 1998). Each docking experiment was performed 20 times, yielding 20 docked conformations. Parameters for the docking are as follows: population size of 150; random starting position and conformation; maximal mutation of 2Å in translation and 50° in rotations; elitism of 1; mutation rate of 0.02 and crossover rate of 0.8; and local search rate of 0.06. Simulations were performed with a maximum of 2.5 million energy evaluations and a maximum of 27 000 generations. Cubic grid, centred on the active site, with 108 points spaced by 0.375 Å was generated using AutoGrid 4 for each structure. VS with docking was performed on a Linux Cluster Platform (INRA MIGALE bioinformatics platform; http://migale.jouy.inra.fr) which contains 160 CPU (80 Intel Quad Core 5340 2.33GHz, 40 Intel Dual Core 5140 2.33GHz and 40 Xeon 3.2GHz) on 40 computing nodes. PDB files used for docking were 2bpw (Chang *et al.*, 2007), 1uwb (Wang *et al.*, 2005) and 1afe (Schafferhans and Klebe, 2001) for HIV-1 protease, HIV-1 reverse transcriptase and thrombin, respectively. The ligand and crystallographic waters were removed. Polar hydrogens were added. Receptors were prepared with Python AutoDockTools scripts. Polar hydrogens were added to small molecules and energy minimization was computed using PRODRG (Schüttelkopf and van Aalten, 2004).

3.3 Contacts analysis

3.3.1 Vectors generation AuPosSOM program processes contacts analysis (hydrogen bonds and hydrophobic contacts) for each docked molecule. Contacts are represented as interleaved vectors. The cardinality of each vector is the double of the number of amino acids in the protein targeted. The hydrogen bonds are computed over all possible donor–acceptor pairs, such that one atom belongs to the protein and the other to the ligand. An interaction is considered as an hydrogen bond when the D-H...A distance is 1.85 ± 0.65Å and the D-H...A angle is $180 \pm 80°$. Interactions are considered as hydrophobic ones when two hydrophobic atoms are closer than 3.9Å. Bio.PDB module (Hamelryck and Manderick, 2003) from Biopython project (http://biopython.org) was used to measure angle and distances from coordinates contained in PDB files of the docked structures obtained. The AuPosSOM performance was tested on 20 000 structures (1000 molecules with 20 poses per molecule). On one Linux PC (quadricore Intel 2.66 GHz, 1GB RAM) this calculation takes about 58 min, or on average 0.174s per structure.

3.3.2 Neural network analysis With vectors described in contacts analysis section, we used a Euclidean SOM, trained in two phases with the following parameters: map size 5×4, exponential decrease of learning rate and radius; phase 1: starting learning rate 0.2, starting radius 6, 1000 iterations; phase 2: starting learning rate 0.02, starting radius 3, 10 000 iterations. Such training length ensured that cluster tree topology remained unchanged with any additional training. Consensus trees were made with different map sizes in order to test cluster confidence. A map size of 20 neurons gives confident clusters. As shown by Samsonova *et al.*, (2006) smaller SOMs tend to yield more confident clusters. We further test that the tree resulting from a randomly initialized map is not dependent on the order of the data elements. Furthermore, the fact to tag molecules as known or unknown does not influence in anyway SOM analysis, but these data are just used for plotting ROC curves.

AuPosSOM distribution includes new python implementations of Kohonen SOM and unsupervised clustering.

3.3.3 k-means clustering In order to compare SOM algorithm with another clustering method, a Python implementation of k-means calculation has been made and included in the AuPosSOM software. The algorithm proceeds by alternating between two steps: (i) assignment step: assign each vector to the cluster with the closest mean (centroid), (ii) update step: calculate the new means (centroids) of the vectors in the cluster. To initialize the process, k initial centroids are randomly chosen from the dataset. In addition, the UPGMA (Unweighted Pair Group Method with Arithmetic mean) algorithm has been written to transform the pairwise distance matrix between clusters into a rooted tree for the results visualization.

4 RESULTS

The ROC plot provides not only a way to fine-tune the parameters of VS processes but also allows a direct comparison of different VS workflows (Triballeau *et al.*, 2005): the closer a curve comes to the upper-left corner of the graph, the better the screening process is.

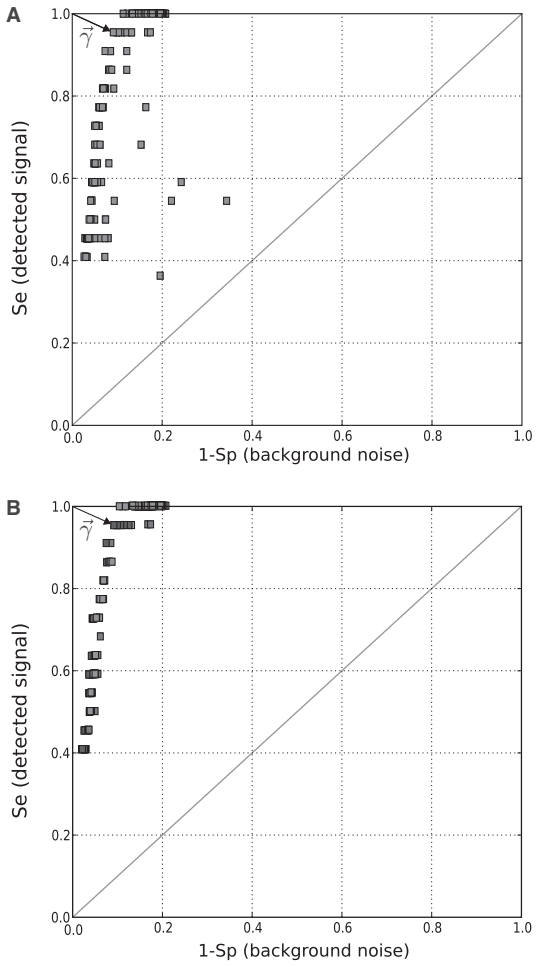


Fig. 2. ROC plots for HIV-1 protease model. For each plot, $\vec{\gamma}$ with minimal norm is represented. (**A**) ROC plot obtained using AuPosSOM software with various subensembles of vectors to train the Kohonen map. The number of elements for each subensemble used is <25% of the total number of molecules in the database. (**B**) Effect of enrichment of the dataset ($\mathbb{L}$ ensemble) used for training the SOM. The number of elements for each subensemble used is >25% of the total number of molecules in the database.

Find out minimal γ factor described in Equation (3) is a way to find out the closest point to the upper-left corner of the graph. Consequently, this minimal γ is a quantification of the performance of the test. The smaller the γ factor is, the better the test is at isolating signal from background noise. Another way to measure the overall performance of the computer test is to calculate the area under the curve (AUC). This calculation is possible for simple tests (e.g. using score-filter), but is much more difficult for ROC plot resulting from neural network analysis. ROC plot obtained from neural network analysis of poses and from different scoring threshold are given in Figure 2 and 3, respectively.

The first observation to be made from these two figures is that the ROC curves remain above the diagonal representing a random distribution. The ROC curve derived from neural network analysis is plotted according to the workflow described in Sections 2 and 3. Points of the curve result from different training sets used for training

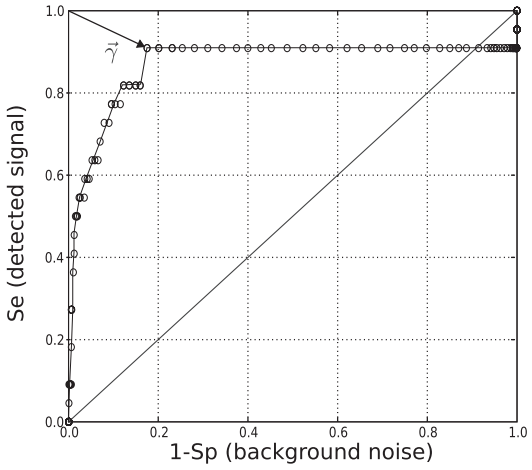


Fig. 3. ROC plot for HIV-1 protease model using the AutoDock 4.0 scoring function with various score thresholds.

the map. Training sets are subensemble of vectors defined as $\mathbb{L}$ containing all vectors of the ensemble $\mathbb{K}$ and a random set of vectors of the ensemble $\mathbb{U}$ (each element of the different ensembles is a vector describing corresponding compound). Effect of enrichment of the training set $\mathbb{L}$ is shown in Figure 2A and B. Figure 2A reports the overall result of the screening process with different sets of training vectors; for each set the number of vectors is <25% of the total ensemble of vectors. Effect of enrichment of the number of training vectors is shown in Figure 2B. The increase of the number of vectors in the training set yields less dispersion in the ROC plot. When the number of training vectors is too small the effect due to the addition of new randomly chosen vectors is important leading to outliers in Figure 2A. All true positive compounds are detected (Se = 1) with a small proportion of false positives ($1 - \text{Sp} = 0.10$). The point with the smallest γ value is not the most sensible point but a point with a sensitivity of 0.95 and a background noise ($1 - \text{Sp}$ value) of 0.09 (Table 1).

Score analysis of the docking is not able to find out all active compounds. This is due to a bad estimation of the ligand binding energies for two ligands. This problem is overcome by the contact analysis approach developed. However, the point with the smallest γ value shows a sensitivity of 0.91 and a background noise of 0.18 (Table 1). The ROC plot obtained with AutoDock 4.0 scoring function with 22 known ligands is in accordance with the result obtained by Chang *et al.* (2007) with 11 known ligands.

ROC analysis for two other models (Fig. 4)—HIV-1 reverse transcriptase and human thrombin—shows that contact clustering gives more sensible and more specific results than score analysis provided by AutoDock 4.0 software. However, results obtained with human thrombin show no significant amelioration. For reverse transcriptase, score analysis is not able to discriminate active compounds. This is either due to a bad estimation of protein ligand binding energy or incorrect poses conformation. Surprisingly, contact clustering analysis is able to find out 10 active compounds among 14 (71%). Therefore, AutoDock 4.0 is able to generate good conformations for HIV-1 reverse transcriptase inhibitors.

For all models tested so far, AuPosSOM analysis yields more sensible and more specific results than score analysis provided by

Table 1. Sensitivity [Se, Equation (1)] and specificity [Sp, Equation (2)] with smallest γ value [Equation (3)] for each model

		SOM clustering			k-means clustering			Score analysis		
Model	card($\mathbb{K}$)	Se	Sp	γ	Se	Sp	γ	Se	Sp	γ
HIV-1 protease (2bpw)	22	0.95	0.91	0.10	0.82	0.94	0.19	0.91	0.82	0.20
HIV-1 reverse transcriptase (1uwb)	14	0.57	0.90	0.44	0.71	0.70	0.42	0.50	0.63	0.62
Human thrombin (1afe)	20	0.85	0.85	0.21	0.80	0.90	0.22	0.80	0.85	0.25

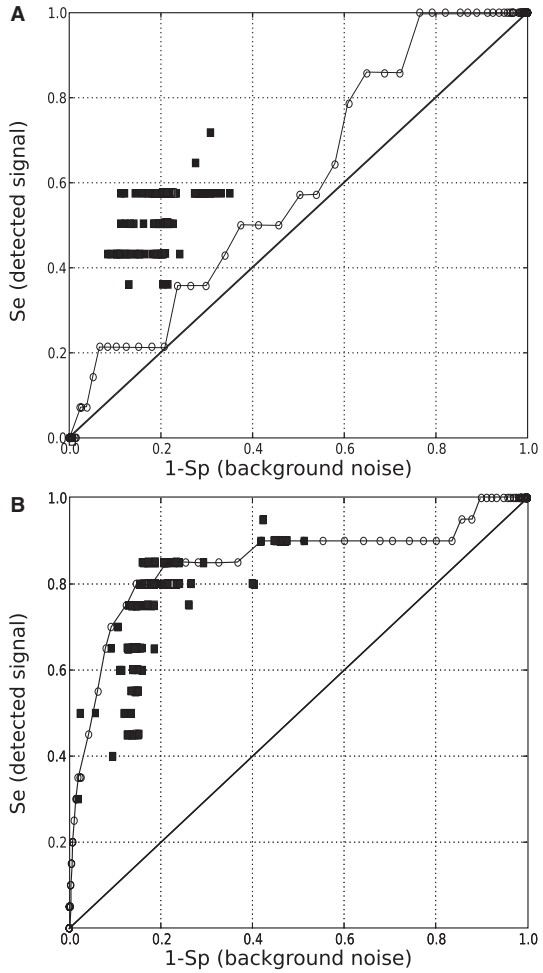


Fig. 4. ROC plots obtained from score analysis using AutoDock 4.0 scoring function (open circle) and using contact analysis using AuPosSOM software (filled points). (**A**) ROC plots obtained from HIV-1 reverse transcriptase screening. (**B**) ROC plots obtained from human thrombin screening.

AutoDock 4.0 package. The resultant γ values are consequently lower (Table 1) for contact analysis processes.

Unrooted tree representation of contact footprints clustering for protease screening experiment with the biggest sensitivity (Se = 1) and the smallest background noise (1 − Sp = 0.1) is given in Figure 5. $\mathbb{E}$ is the ensemble with the smallest γ value [see Equation (3)]. A leave is defined as a subensemble of elements (compounds of the database screened) belonging to $\mathbb{K}$ or $\mathbb{U}$. A tree is composed of three different leaves: (i) leaves composed of U elements exclusively, notated as $\{U\}_{U\in\mathbb{U}}$, (ii) leaves composed of K elements exclusively, notated as $\{K\}_{K\in\mathbb{K}}$ and (iii) leaves composed of K and U elements, notated as $\{K\}_{K\in\mathbb{K}\cup\{U\}_{U\in\mathbb{U}}}$. Branch length is proportional to divergence of contacts made by molecules belonging to the different leaves. The first observation to be made from this tree is that the ensemble $\mathbb{E}$ is isolated from the rest of the tree ($\mathbb{U}\setminus\mathbb{E}$ ensemble). This means that the Kohonen neural network found that compounds which belong to $\mathbb{E}$ interact with their target with distinct contact patterns. Thus, active compounds can be isolated from inactive ones using only mean protein contact footprints. This result demonstrates the existence of a contact–activity relationship (CAR). Furthermore, this CAR can be employed as a new VS process.

A comparison between the SOM clustering and a baseline clustering method such as k-means has been done for the three models. Due to a random initialization for both SOM and k-means clusterings, different results can be obtained with the same set of parameters. Furthermore, the k value or the map size influences the results. ROC calculations were performed 10 times, with k variations ($k=\{2,\ldots,30\}$) for k-means method or map size variations ($(X,Y)=\{(i,j)\in\mathbb{N}^2 : 2\leq i\leq j\leq 6\}$) for SOM. The performance of the SOM network is higher for the HIV-1 protease model (Fig. 6): SOM clustering method yields more sensible results (Se = 1.00) than a simpler method such as k-means (Se = 0.82). SOM clustering method leads to less dispersion for sensitivity and specificity than k-means algorithm. According to γ values (Table 1), no significant amelioration is brought by the SOM clustering for HIV-1 reverse transcriptase and human thrombin in comparison with the k-means method. However, the ROC plots obtained for HIV-1 reverse transcriptase and human thrombin (See Figs S1 and S2 in Supplementary Material) show less dispersion for SOM clustering.

To understand these results, dispersion of the input data has been studied. Shannon's entropy has been calculated (Supplementary Material) for all the vectors for the three models and results reported in a histogram (see Fig. S3 in Supplementary Material). The dispersion calculated for protease data ($\bar{\rho}=0.84$; $\sigma=0.04$, defined in Supplementary Material) is more important than the dispersion of the data for reverse transcriptase ($\bar{\rho}=0.93$; $\sigma=0.02$) and thrombin ($\bar{\rho}=0.93$; $\sigma=0.01$). $\bar{\rho}$ value gives the mean dispersion within the vectors. σ value gives the mean dispersion between vectors. As the level of dispersion in the data increases, the performance advantage of the SOM network relative to the k-means clustering method increases to a dominant level.

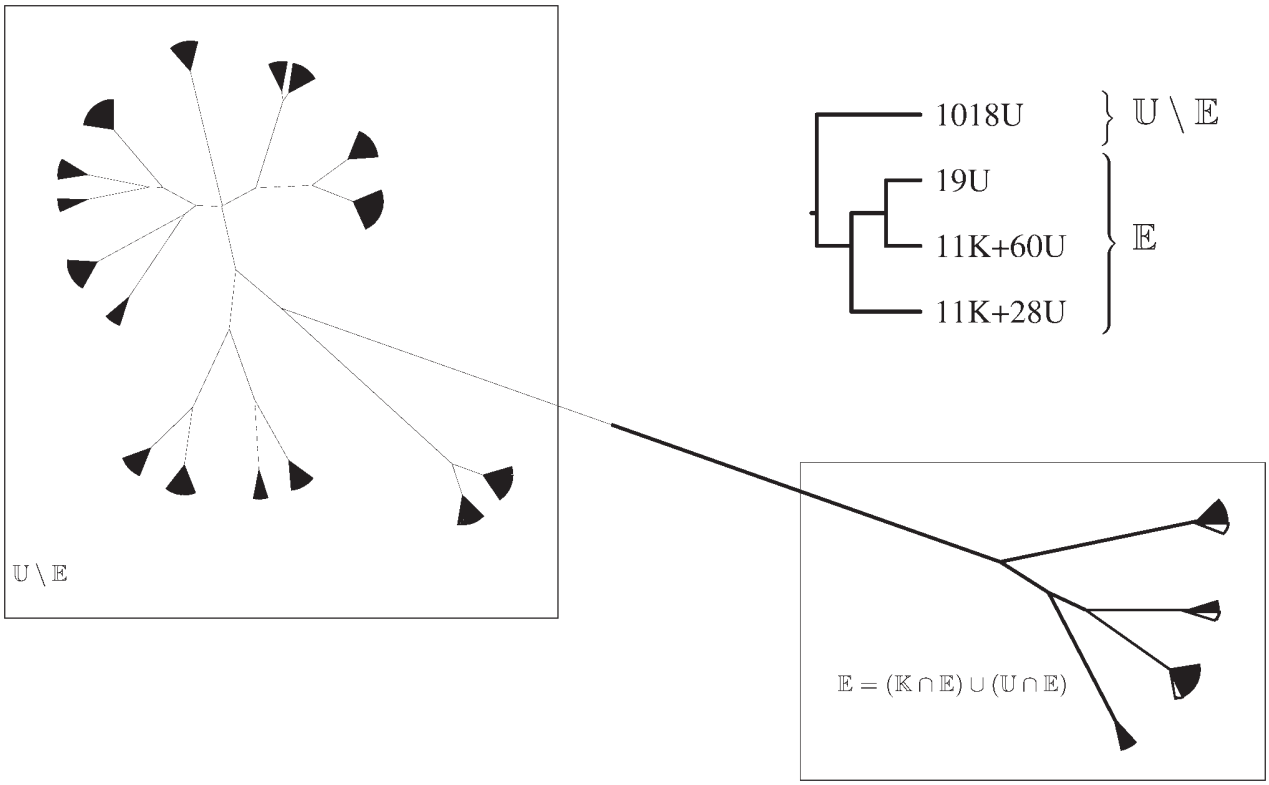


Fig. 5. Tree representation of contact footprints clustering for protease screening experiment. The tips stand for the different molecules of the database screened. Branch length is proportional to divergence of contacts made by ligands belonging to the different clusters (leaves). Molecules with known activities (*K* compounds) are represented with white tips. The ensemble $\mathbb{E}$ with the smallest γ value according to the ROC plot is shown in bold. A simplified tree (inset) shows the number of *K* compounds and *U* compounds in homogeneous leaves ensembles.

In addition, important structural informations are found in this analysis of atomic interactions. Contact clustering allows identification of key residues implicated in ligand binding. These residues can be mapped onto the target structure. For example, with the HIV-1 protease (Fig. 7), a clear relationship is found between footprints of the ensemble $\mathbb{E}$ and amino acids thought to be essential to the interaction of antiproteases. Major point mutations associated with resistance to protease inhibitors (Johnson *et al.*, 2005) and located in the binding site correlates with the contact pattern characteristic of compounds found to be active by the automatic contact analysis. Only one amino acid, namely Ala-28, is found to make essential interaction without point mutation at this position. Interestingly, Ala-28 is highly conserved and important for the structuration of HIV-1 protease. This amino acid can be exploited to design resistance-evading drugs (Wang and Kollman, 2001).

AuPosSOM analysis can be performed with docking results given by other docking programs. This strategy has been employed for a VS process to search new β-TrCP inhibitors with a docking protocol described by Evrard-Todeschi *et al.* (2008) using the Surflex-Dock 2.0 program (Jain, 2003) from Sybyl 7.3 molecular modelling program package (Tripos Inc., France) (data not shown).

Docking on HIV-1 protease has also been performed using Surflex-Dock 2.0 with the same dataset presented in Section 3. Score analysis with Surflex-Dock 2.0 scoring function gives more sensible results than thus obtained with AutoDock 4.0 program. However, AuPosSOM analysis on docking poses proposed by Surflex–Dock yields more specific results than Surflex–Dock scoring (see docking procedure and Fig. S4 in the Supplementary Material).

5 DISCUSSION

Incorrect estimation of ligand–protein free energy of binding leads to the apparition of false negative and even false positive compounds in score-based VS process. Chang *et al.* (2008) observed that incorrect lowest energy conformations will be found in ∼1% of docking experiments and the correct conformation will be found in 25–100% of the experiments. Thus, a simple procedure that chooses the conformation of best energy from a set of multiple docking experiments will yield an incorrect conformation. With the process exposed here, all poses per molecule are processed since mean vectors for all poses are calculated. Thus, distribution of the digits in the resulting vectors gives information about poses cluster size, and conformation frequencies: amino acids which are contacted with high frequency for the 20 poses calculated for a ligand are represented in the resulting vector by a higher figure than the other ones. As the frequency of finding a given conformation is providing information on the energy landscape of binding, and that a high frequency is a measure of favourable entropy in the binding process (Chang *et al.*, 2008), our statistical

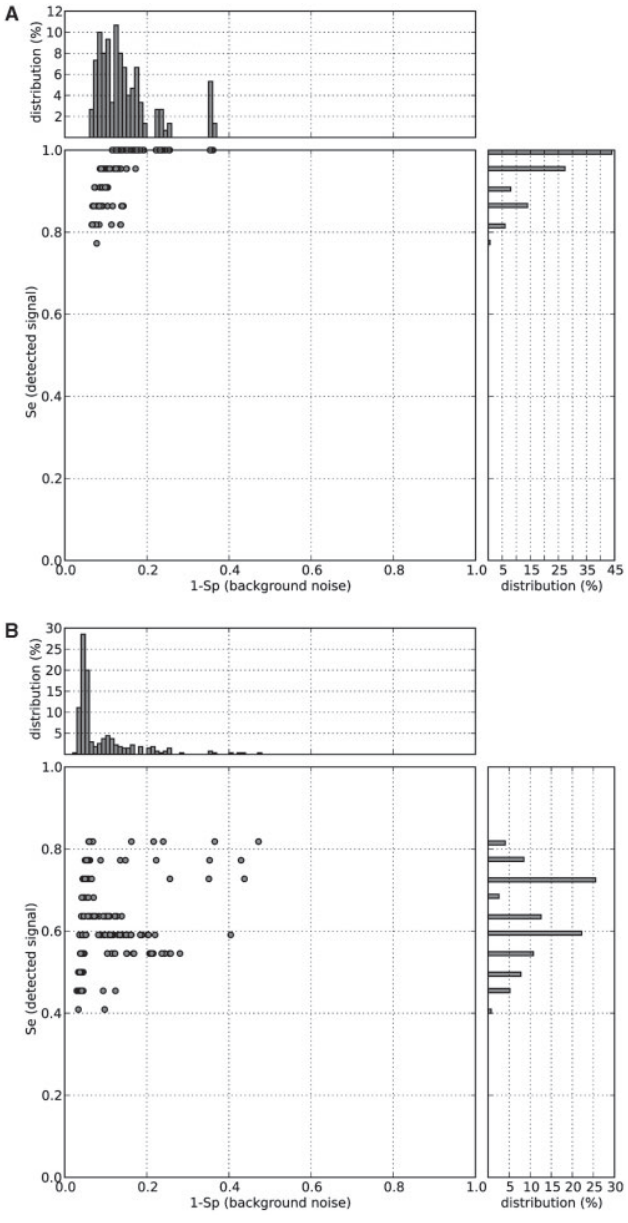


Fig. 6. ROC plots obtained from (**A**) Kohonen SOM and (**B**) *k*-means clustering methods for different Kohonen map sizes and different *k*-values for the HIV-1 protease dataset. For each map size and each *k*-values, calculations are repeated 10 times. The histograms show points distribution. The values of the most represented point are: Se = 1, Sp = 0.87and Se = 0.73, Sp = 0.95 for SOM and *k*-means, respectively.

analysis of poses takes into account the vibrational entropy factor.

TreeSOM algorithm developed by Samsonova *et al.* (2006) is an unsupervised method for cluster analysis and confidence testing for SOMs. This process allows to find reliable clusters. Interestingly, learning samples influence final clustering. Iteration over learning samples coupled with ROC plot, allows to find out the most appropriate dataset to solve classification problem between active

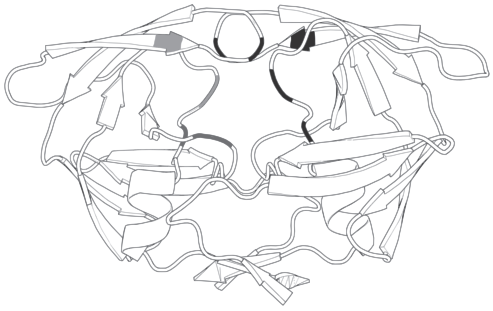


Fig. 7. HIV-1 protease structure. Left monomer shows contact footprints characteristic of the ensemble $\mathbb{E}$. Greyscale gradient indicates contact frequency. Right monomer highlights point mutations associated with HIV-1 protease resistance to inhibitor (Johnson *et al.*, 2005).

compounds and inactive ones. However, default learning with all vectors gives good results for the three models tested. Furthermore, active compounds are isolated from the other leaves of the tree without taking into account experimental data such as key residues or activities. Thus, good classification can be obtained from default run without plotting ROC curves.

k-means clustering (implemented in AuPosSOM) could give interesting results when input data are less dispersed. Compared with *k*-means clustering, SOM clustering is more efficient when input data are dispersed (e.g. for the protease dataset) (Mangiameli *et al.*, 1996). Furthermore, the performance of the SOM network is shown to be robust across a wide range of data imperfection inherent to outlier poses proposed by the docking software. This advantage is explained by the iterative learning of the Kohonen map which may neutralize the noise produced by incorrect poses of docking. Moreover, SOM network is less sensitive to its set of parameters, and cluster results are therefore more repeatable.

Another approach based on structural interaction fingerprints (SIFt) has been described to analyse and organize protein-small molecule complexes (Deng *et al.*, 2004). In this method, similarity measurement between vectors is based on the Tanimoto coefficient and a hierarchical clustering is done using an agglomerative hierarchical clustering. Compared with the method proposed in this article, this process has the same advantages: it allows an unbiased docking protocol and a bypass to scoring functions. Furthermore, interaction descriptor-based methods enable scaffold-hopping. However, the use of the Tanimoto coefficient as a similarity measurement leads to two major drawbacks: (i) due to slight errors in the binding geometry, missing components in interaction vectors may lead to incorrect clustering; (ii) comparison of macro-ligands (e.g. peptide) and small organic compounds could yield locally similar patterns but globally different vectors, which results in high Tanimoto coefficient whereas local binding mode may be the same. Furthermore, SIFt method needs known complex structures as referenced fingerprints, thus only expected binding modes are selected. Finally, one correct pose must be selected, according to the reference, which sets aside the statistical entropic analysis.

In order to check the influence of the number of known active compounds included in the database for the benchmarks presented here, the probability of clustering at least all active compounds (true positives) with an undefined number of false positive was calculated.

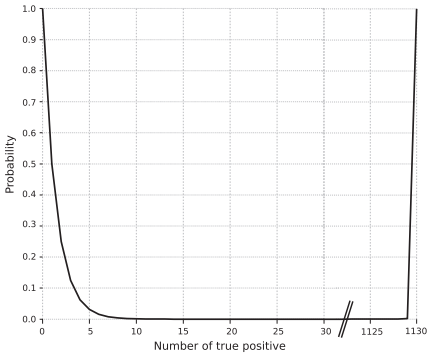


Fig. 8. Effect of the number of true positive in a database of 1130 compounds in the probability of clustering at least all active compounds [see Equation (4)].

This probability P is given by :

$$P = \frac{\sum_{i=0}^{n-p} C_{n-p}^{i}}{\sum_{j=p}^{n} C_{n}^{j}} \qquad (4)$$

where p is the number of true positive and n the total number of compounds. For the HIV-1 protease screening, $p = 22$ and $n = 1130$. Thus, the probability P of clustering at least all active compounds is $2.38 \cdot 10^{-7}$. For a database composed of 1130 molecules, the probability P <1% when the number of true positive is more than 6. This probability becomes very small ($P < 10^{-3}$) when the number of true positive is more than 10 (Fig. 8). Therefore, the number of active compounds put into the total number of compounds of the database is a good benchmark to check the reliability of the contact clustering process presented in this article.

From a dataset of compounds, contact clustering is able to discriminate active compounds based only on CAR. We observed that using first this analysis of the position of the docked molecules is more successful to find out active compounds than a classical energy analysis. This new method can be integrated into VS process currently available to rapidly visualize results. Furthermore, contact clustering process allows identification of key residues implicated in ligand binding. Hierarchical classification of compounds according to their contacts can be employed as a starting point to achieve *in silico* drug-design process.

ACKNOWLEDGEMENTS

We are grateful to the INRA MIGALE bioinformatics platform (http://migale.jouy.inra.fr) for providing computational resources. We thank Dr Daniele Marinazzo from the Laboratory of Neurophysics and Physiology, CNRS UMR 8119, at Université Paris Descartes for his helpful discussions on the clustering methods.

Conflict of Interest: none declared.

REFERENCES

Alvarez,J. and Shoichet,B. (2005) *Virtual Screening In Drug Discovery*. CRC Press, Boca Raton, USA.

Chang,M. *et al.* (2007)Analysis of HIV wild-type and mutant structures via in silico docking against diverse ligand libraries. *J. Chem. Inf. Model.*, **47**, 1258–1262.

Chang,M. *et al.* (2008)Empirical entropic contributions in computational docking: evaluation in APS reductase complexes. *J. Comput. Chem.*, **29**, 1753–1761.

Deng,Z. *et al.* (2004) Structural interaction fingerprint (SIFt): a novel method for analyzing three-dimensional protein-ligand binding interactions. *J. Med. Chem.*, **47**, 337–344.

Evrard-Todeschi,N. *et al.* (2008) Structure of the complex between phosphorylated substrates and the SCF β-TrCP ubiquitin ligase receptor: a combined NMR, molecular modeling, and docking approach. *J. Chem. Inf. Model.*, **48**, 2350–2361.

Hamelryck,T. and Manderick,B. (2003)PDB file parser and structure class implemented in Python. *Bioinformatics*, **19**, 2308–2310.

Huey,R. *et al.* (2007) A semiempirical free energy force field with charge-based desolvation. *J. Comput. Chem.*, **28**, 1145–1152.

Jain,A. (2003) Surflex: fully automatic flexible molecular docking using a molecular similarity-based search engine. *J. Med. Chem.*, **46**, 499–511.

Johnson,V. *et al.* (2005) Update of the drug resistance mutations in HIV-1: fall 2005. *Top HIV Med*., **13**, 125–131.

Kitchen,D.B. *et al.* (2004) Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat. Rev. Drug Discov.*, **3**, 935–949.

Kohonen,T. (2001) *Self-Organizing Maps*. Springer Series in Information Sciences, Heidelberg, Germany.

Liu,T. *et al.* (2007)BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Res.*, **35**, D198–D201.

Mangiameli,P. *et al.* (1996) A comparison of SOM neural network and hierarchical clustering methods. *Eur. J. Oper. Res.*, **93**, 402–417.

Morris,G. *et al.* (1998)Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *J. Comput. Chem.*, **19**, 1639–1662.

Muegge,I. (2008) Synergies of virtual screening approaches. *Mini Rev. Med. Chem.*, **8**, 927–933.

Renner,S. *et al.* (2008)Maximum common binding modes (MCBM): consensus docking scoring using multiple ligand information and interaction fingerprints. *J. Chem. Inf. Model.*, **48**, 319–332.

Samsonova,E.V. *et al.* (2006) TreeSOM: cluster analysis in the self-organizing map. *Neural Netw.*, **19**, 935–949.

Schafferhans,A. and Klebe,G. (2001) Docking ligands onto binding site representations derived from proteins built by homology modelling. *J. Mol. Biol.*, **307**, 407–427.

Schüttelkopf,A. and van Aalten,D. (2004) PRODRG: a tool for high-throughput crystallography of protein-ligand complexes. *Acta Cryst.*, **60**, 1355–1363.

Triballeau,N. *et al.* (2005)Virtual screening workflow development guided by the "receiver operating characteristic" curve approach. Application to high-throughput docking on metabotropic glutamate receptor subtype 4. *J. Med. Chem.*, **48**, 2534–2547.

Wang,J. *et al.* (2005)Hierarchical database screenings for HIV-1 reverse transcriptase using a pharmacophore model, rigid docking, solvation docking, and MMPB/SA. *J. Med. Chem.*, **48**, 2432–2444.

Wang,W. and Kollman,P. (2001)Computational study of protein specificity: the molecular basis of HIV-1 protease drug resistance. *Proc. Natl Acad. Sci. USA*, **98**, 14937–14942.

Warren,G.L. *et al.* (2006) A critical assessment of docking programs and scoring functions. *J. Med. Chem.*, **49**, 5912–5931.

Waszkowycz,B. (2008) Towards improving compound selection in structure-based virtual screening. *Drug Discov. Today*, **13**, 219–226.

Yang,J. *et al.* (2005)Consensus scoring criteria for improving enrichment in virtual screening. *J. Chem. Inf. Model.*, **45**, 1134–1146.

6.2 Criblage de la DUD : *A Directory of Useful Decoys*

Afin de valider l'outil AuPosSOM dans le domaine du criblage virtuel il est nécessaire d'évaluer celui-ci en utilisant une autre chimiothèque. Afin que cette validation soit reconnue par la communauté scientifique dans le domaine et qu'elle soit comparable à d'autres tests il est nécessaire d'utiliser une chimiothèque accessible publiquement et contenant un ensemble de molécules difficiles à discriminer comme actives ou inactives. De plus l'annotation des molécules doit être fiable et complète afin que les molécules actives soient réellement actives contre la cible considérée et que les molécules inactives soient de véritables leurres (*decoys*) c'est à dire des molécules ne se liant pas sur cette même cible. Ceci permet d'obtenir des courbes de ROC fiables et objectives, basées sur des résultats expérimentaux et non des hypothèses concernant l'activité des molécules.

La chimiothèque DUD (*a Directory of Useful Decoys*) a été créé dans cette optique et est présentée en détails dans la partie 3.2.1 page 56.

Pour des raisons de temps, l'ensemble de la chimiothèque, qui est composée de 2 950 composés actifs sur 40 cibles protéiques et de 106 200 leurres, n'a pu être testé.

La transcriptase inverse du VIH-1 a été sélectionnée comme cible pour tester le logiciel et ceci pour deux raisons :

- les résultats du criblage avec la chimiothèque du laboratoire sur la transcriptase inverse n'ont pas été à la hauteur des résultats escomptés (voir figure 6.6 page 122),
- la mise au point d'un protocole de criblage virtuel sur la transcriptase inverse du VIH-1 est intéressante afin de tester sur cette cible des chimiothèques privées de la société CellVir SAS.

La transcriptase inverse du VIH-1 est une cible importante de la thérapie antivirale. La thérapie HAART inhibe la réplication du VIH par l'administration de plusieurs types de molécules actives : des inhibiteurs nucléosidiques (NRTIs), nucléotidiques (NtRTIs), non nucléosidiques (NNRTI) de la transcriptase inverse et des inhibiteurs de protéase [8]. L'ensemble du criblage réalisé porte sur les molécules du type NNRTI.

Bien que extrêmement intéressante pour le développement de molécules anti-rétrovirales la transcriptase inverse reste une cible difficile pour les études par modélisation moléculaire. En effet cette protéine présente une importante flexibilité conformationnelle qui complique l'étude par *docking*. La poche de fixation de ces molécules subit d'importantes modifications conformationnelles lors de la fixation de la molécule [72].

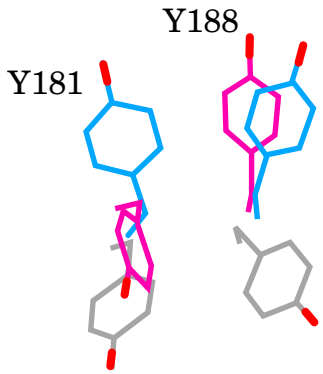


FIGURE 6.11 – Superpositions de trois structures cristallographiques montrant les différentes orientations possibles des tyrosine 181 et 188 de la poche de fixation des NNRTIs. La structure en cyan (code pdb : 1rt4), montre l'orientation canonique pour la plupart des complexes entre la transcriptase inverse et un NNRTI. La forme *apo* (code pdb : 1rtj [41]) est en gris. La structure en magenta (code pdb : 2be2) montre l'orientation rare de la tyrosine 181 sous la forme *down* pour un complexe entre la transcriptase inverse et une molécule NNRTI.

La comparaison de la structure de la transcriptase inverse seule (forme *apo*) avec plus de soixante-dix structures de transcriptases inverses en interaction avec des NNRTIs montre une importante modification conformationnelle des tyrosines 181 et 188. La forme apo est caractérisée par les tyrosines 181 et 188 pointant vers le bas (*down*) tandis que les structures en interaction avec des NNRTIs ont ces mêmes tyrosines qui pointent vers le haut (*up*). Cependant il existe six structures cristallographiques (1fko [147], 1rt3 [146], 1rti [145], 1tv6 [139], 2b5j [68] et 2be2 [68]) de complexes avec des NNRTIs où la tyrosine 181 est sous la forme *down*, dans la même position que la forme *apo* (figure 6.11).

Ainsi, le choix du récepteur est primordial dans la réalisation du *docking*. Afin d'échantillonner l'ensemble des conformations possible de la poche de fixation de la transcriptase inverse trois structures ont été étudiées en *docking* : 2be2, 1jla et 1rt4. La structure 1rt4 représente la forme canonique de la conformation de la poche de fixation en complexe avec un NNRTI (en cyan sur la figure 6.11). La structure 2be2 représente une conformation alternative de la poche (en magenta sur la figure 6.11). Finalement la structure 1jla possède une mutation de la tyrosine 181 en cystéine (Y181C). Cette mutation apparaît rapidement suite à un traitement utilisant des NNRTIs et entraîne une résistance à la plupart d'entre eux [47]. Cette structure a été

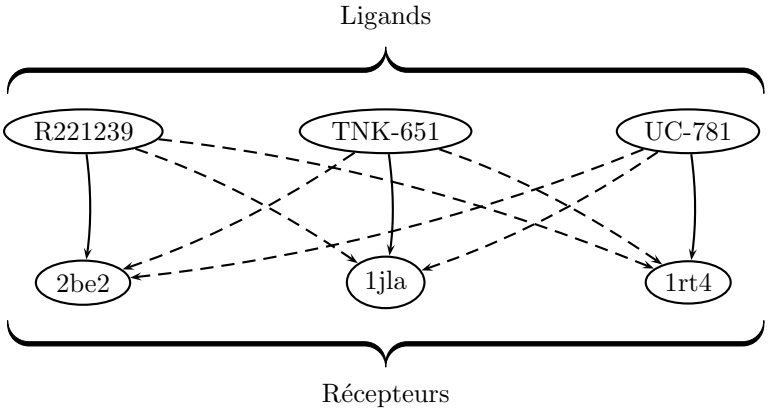


FIGURE 6.12 – Ce diagramme représente les neuf calculs de *docking* réalisés sur les structures 2be2, 1jla et 1rt4 de la transcriptase inverse du VIH-1. Les lignes pleines indiquent les *dockings* réalisés sur les coordonnées atomiques du récepteur obtenues avec le ligand considéré (*docking* direct). Les lignes pointillées indiquent des calculs de *docking* sur les coordonnées atomiques d'un récepteur en interaction avec un autre ligand (*docking* croisé).

utilisée dans des expériences de criblage virtuel afin de trouver de nouveaux inhibiteurs échappant à ce phénomène de résistance [134]. Les trois structures étudiées proviennent de données de diffractions des rayons X effectuées sur un co-cristal transcriptase inverse / NNRTI. Ces inhibiteurs sont R221239, TNK-651 et UC-781 pour les structures 2be2, 1jla et 1rt4, respectivement.

Les *dockings* directs et croisés, soit un total de neuf calculs, ont été réalisés comme expliqué sur la figure 6.12.

Ces différents *dockings* ont été réalisés avec le logiciel Dock 6.3 et vingt poses par calcul ont été générées. La fonction de score implémentée dans le logiciel Dock a premièrement été utilisée afin d'évaluer les énergies d'interaction entre ligand et protéine (voir tableau 6.2 page ci-contre).

La racine carrée de la déviation moyenne au carré (*root mean square deviation* ; RMSD) a aussi été calculée entre la pose de plus basse énergie pour le ligand considéré et les coordonnées atomiques du ligand dans la structure cristallographique correspondante. Ces résultats sont présentés dans le tableau 6.2 page suivante.

Pour le ligand R221239 la structure la plus proche de la structure cristallographique, en terme de RMSD, est obtenue par *docking* direct. La structure de plus basse énergie est aussi celle obtenue par *docking* direct. Pour le ligand TNK-651 la pose de *docking* la plus proche des données expérimentales est obtenue par *docking* croisée sur la structure 2be2. Ce n'est pas la structure de

	RMSD (Å)	Score Dock (kcal · mol^{-1})	RMSD	Score Dock	RMSD	Score Dock
	2be2		1jla		1rt4	
R221239	0,60	-66,43	1,16	-54,62	2,34	-40,14
TNK-651	1,05	-62,94	1,38	-63,55	1,62	-45,50
UC-781	2,15	-52,95	2,84	-51,67	0,78	-54,76

TABLE 6.2 – RMSD entre les coordonnées atomiques des ligands dockés et les coordonnées atomiques des ligands correspondant dans la structure cristallographique, ainsi que le score de *docking* obtenu avec la fonction de score implémentée dans Dock 6.3.

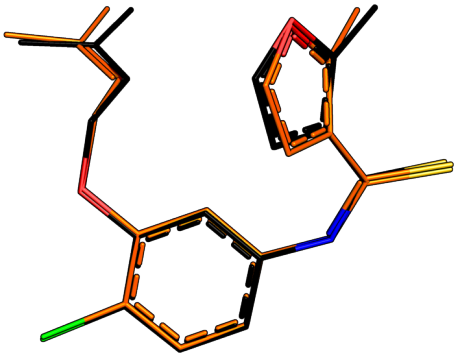
plus basse énergie, celle ci étant obtenue par *docking* direct, mais la différence d'énergie n'est pas significative ($\Delta E = 0,61\,\text{kcal} \cdot \text{mol}^{-1}$). Ce résultat peut s'expliquer par le fait que la structure 1jla a été résolue afin de comprendre les mécanismes de résistance induite par la mutation Y181C. Cette mutation entraîne une perte de la stabilisation du ligand par interaction entre le cycle aromatique de la tyrosine 181 et un cycle aromatique de l'inhibiteur [148] ce qui induit une résistance à ce composé. Cependant la structure cristallographique du complexe avec le TNK-651 montre peu de différence avec la structure cristallographique de ce même ligand complexé à la protéine non mutée (code pdb : 1rt2 [72]). La conformation de la tyrosine 181 est *up* dans la structure 1rt2 mais le *docking* du TNK-651 sur la structure 2be2 en conformation *down* est plus proche de la structure du ligand en interaction avec la forme mutée Y181C (1jla). Enfin pour le ligand UC-781 le *docking* direct est plus proche des données cristallographiques que les *dockings* croisés. La structure la plus stable est obtenue pour le *docking* direct.

Les structures obtenues par *dockings* directs sont représentées en superposition avec les structures cristallographiques correspondantes sur la figure 6.13 page suivante. Les différences entre conformations dockées et conformations expérimentales sont minimes. Pour les molécules R221239 et TNK-651 la différence résulte surtout dans la position du *linker* souple.

Afin de tester l'efficacité de ces trois modèles à trouver dans une base de données des molécules actives, trois criblages virtuels ont été lancés en

(a) R221239

(b) TNK-651



(c) UC-781

FIGURE 6.13 – Conformations des ligands R221239, TNK-651 et UC-781 au sein de la poche de fixation de la transcriptase inverse du VIH-1. Sont superposées les structures dockées – *docking* direct – (en orange) avec les structures cristallographiques correspondantes (en noir).

		Structures			
Logiciel de *docking*	Méthode de crible	2be2	1jla	1rt4	1rt1
Dock 6.3	Grid Score	0,47	0,46	0,55	0,42
	Amber Score	n.d.	n.d.	0,49	n.d.
	AuPosSOM	n.d.	n.d.	0,76	n.d.
	k-means $k = 10$	n.d.	n.d.	0,67	n.d.
	k-means $k = 20$	n.d.	n.d.	0,71	n.d.
Surflex Dock	Surflex Score	n.d.	n.d.	0,58	n.d.
	AuPosSOM	n.d.	n.d.	0,87	n.d.
	G_Score	n.d.	n.d.	0,85	n.d.
	PMF_Score	n.d.	n.d.	0,60	n.d.
	D_Score	n.d.	n.d.	0,67	n.d.
	ChemScore	n.d.	n.d.	0,77	n.d.

TABLE 6.3 – AUC calculées suivant les filtres employés.

parallèle. La base de donnée testée a été extraite de la DUD et est composée de 43 ligands à activité anti transcriptase inverse et de 841 molécules leurres (*decoys*). La structure 1rt1 [72], fournie par la DUD, a également été testée.

Les résultats du criblage sont présentés sur la figure 6.14 page suivante sous forme de courbes de ROC pour les quatre structures étudiées. Globalement, la sélection en utilisant la fonction de score implémentée dans le logiciel Dock 6.3 est mauvaise. Pour les quatre structures testées la sélection des composés repose sur un processus quasi-aléatoire. De plus pour les modèles reposant sur les structures 2be2 (figure 6.14 page suivante A), 1jla (figure 6.14 page suivante B) et 1rt1 (figure 6.14 page suivante D) la sélection devient inverse à partir d'un seuil de score. Ceci se reflète par une courbe de ROC qui passe en dessous de la diagonale. Le tableau 6.3 donne le résultat du calcul des AUC pour les quatre courbes de ROC présentées sur la figure 6.14 page suivante. Les AUC calculées pour les structures 2be2, 1jla et 1rt1 sont inférieures à 0,5. Ces structures ne permettent donc pas avec le protocole utilisé de sélectionner des molécules actives dans une chimiothèque. La structure 1rt4 donne les meilleurs résultats en tant que courbe de ROC. Ainsi cette structure a été utilisée par la suite pour améliorer le protocole de criblage

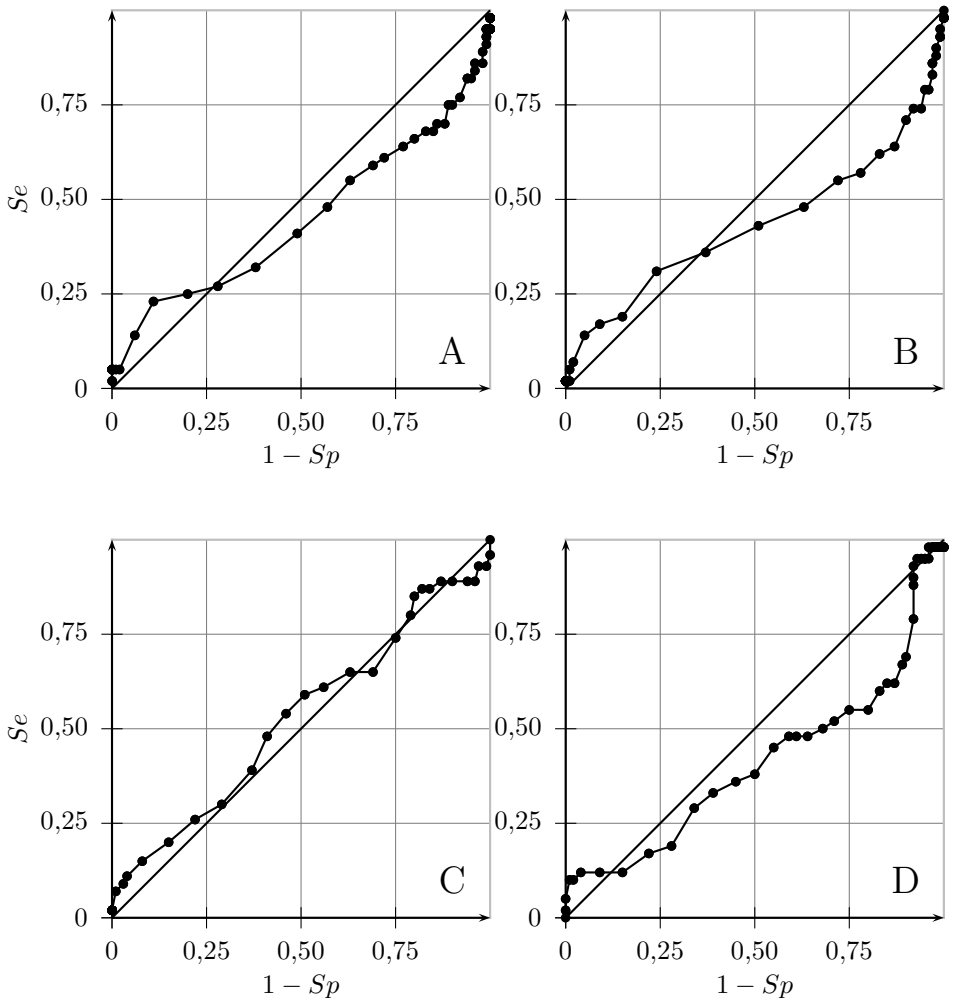


FIGURE 6.14 – Courbes de ROC obtenues après un filtre de score en utilisant le logiciel Dock 6.3 et sa propre fonction de score, sur les cibles 2be2 (**A**), 1jla (**B**), 1rt4 (**C**) et 1rt1 (**D**).

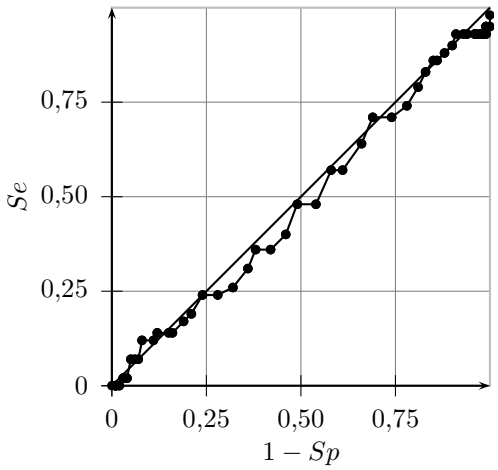


FIGURE 6.15 – Courbe de ROC obtenue par affinement, avec flexibilité du récepteur (code pdb : 1rt4) en utilisant le champ de force AMBER, des poses de *docking* générées par Dock 6.3.

virtuel.

Du fait de la flexibilité importante de la poche de fixation de la transcriptase inverse, un affinement des poses de *docking* avec flexibilité partielle du récepteur (code pdb : 1rt4) basé sur le champ de force AMBER a été réalisé. Cette technique de *post-processing* est décrite dans le paragraphe 3.1.6 page 54. La courbe de ROC obtenue est présentée sur la figure 6.15, l'AUC correspondante est donnée dans le tableau 6.3 page 143. L'AUC passe de 0,55 à 0,49 lorsque la flexibilité du récepteur est pris en compte. La flexibilité du récepteur entraîne une augmentation du taux de faux positifs. Ceci est dû à une mauvaise évaluation des énergies d'interaction et au lissage des différences énergétiques entre ligands et molécules leurres du fait de l'adaptation excessive de la protéine à ces dernières. La flexibilité du récepteur n'est pas facile à prendre en compte pour un processus de criblage où le but est de sélectionner des molécules et donc de comparer des énergies entre différentes molécules. Ce problème est différent lors de l'utilisation de l'outil de *docking* dans la prévision de la conformation du ligand lié. Dans ce cas la prise en compte de la flexibilité peut permettre de générer des conformations plus proches des conformations déterminées expérimentalement.

L'utilisation de ces deux fonctions de score n'étant pas efficace dans la sélection de molécules actives au sein de la chimiothèque, l'analyse des contacts suivi du regroupement des molécules à partir de ceux ci ont été réalisés par le logiciel AuPosSOM. Le résultat obtenu est présenté sous forme de courbe de ROC sur la figure 6.16 page suivante.L'AUC correspondante

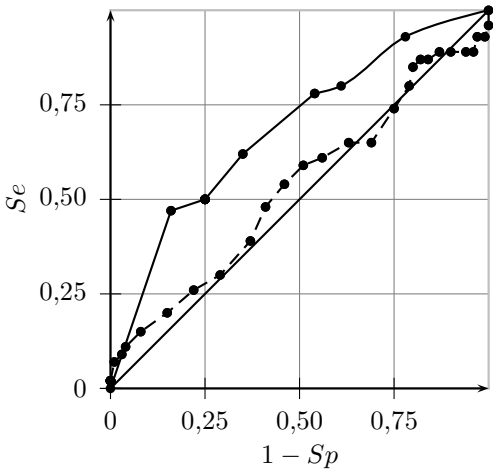


FIGURE 6.16 – Comparaison du filtre de score utilisant la fonction de score implémentée dans Dock 6.3 (Grid Score) (ligne pointillée) et l'analyse de contacts avec le logiciel AuPosSOM (ligne pleine).

est donnée dans le tableau 6.3 page 143. Le regroupement des molécules par analogie de contacts permet d'augmenter la sensibilité et la spécificité du criblage. L'AUC passe de 0,55 pour l'analyse avec la fonction Grid Score à 0,76 avec l'analyse de contact. La courbe de ROC tracée sur la figure 6.16 est obtenue avec un apprentissage avec l'ensemble des molécules de la chimiothèque sur une carte de Kohonen de dimension 5×4. Les coordonnées des points composant cette courbe sont déterminées en calculant pour chaque neurone et ensemble de neurones, suivant le seuil T considéré (voir paragraphe 4.2.4 page 83), la sensibilité et la spécificité. Ainsi le nombre de points obtenus est inférieur au nombre de neurones de la carte, ici vingt, et donc inférieur au nombre de points obtenus pour la courbe de ROC tracée à partir du tri par score de *docking*. Le calcul de l'AUC dépendant du nombre de points de la courbe et sa précision augmentant avec l'augmentation de celui ci, la comparaison des AUC est à faire avec précaution.

Une optimisation de l'apprentissage de la carte de Kohonen selon le protocole décrit sur la figure 6.1 page 114 a été réalisée. Le résultat de l'optimisation est présenté sur la figure 6.17 page suivante. Le gain obtenu par un apprentissage utilisant une sous-chimiothèque de la chimiothèque initiale n'est pas significatif pour la cible et la chimiothèque considérée. Ainsi un calcul utilisant l'ensemble des données donne de suite le résultat optimum.

L'algorithme *k-means* a aussi été appliqué à ce criblage. Les résultats sont présentés sur la figure 6.18 page 148 en comparaison avec l'algorithme

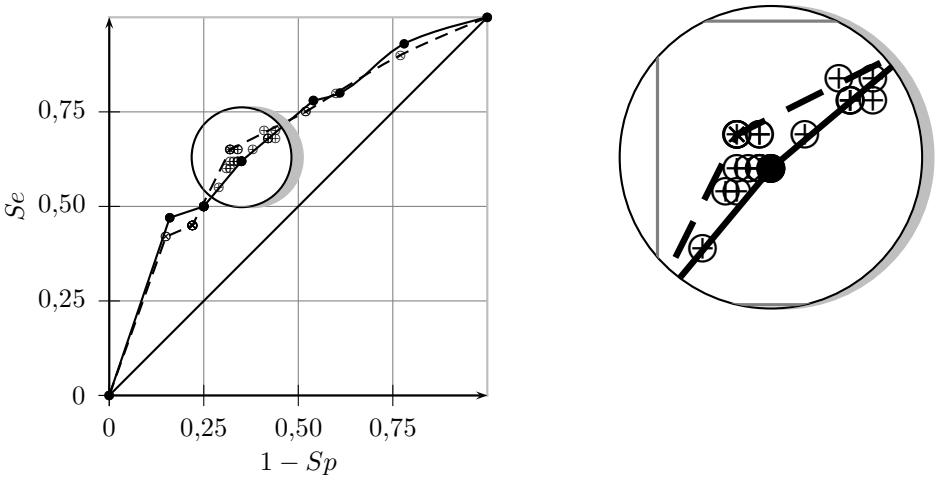


FIGURE 6.17 – Optimisation de l'apprentissage (⊕) de la carte de Kohonen. La ligne pleine correspond au protocole par défaut utilisant l'ensemble des vecteurs d'entrée. La ligne pointillée correspond à un apprentissage avec le sous-ensemble de vecteurs donnant le coefficient γ le plus faible. La figure de droite est un agrandissement de la zone indiquée par le cercle sur la figure de gauche.

SOM utilisé avec les paramètres par défaut. Les AUC correspondantes sont reportées dans le tableau 6.3 page 143. L'algorithme *k-means* n'améliore pas la sensibilité et la spécificité du criblage. L'utilisation des cartes de Kohonen donne de meilleurs résultats que l'algorithme *k-means* pour la cible considérée.

L'influence des poses de *docking* jouant un rôle important dans l'analyse des contacts et les différences importantes existant entre les différentes fonctions de score (voir partie 3.1 page 41) nous ont conduit à tester un autre logiciel de *docking* pour la structure 1rt4. Le logiciel Surflex Dock a été utilisé pour réaliser cette étude. L'utilisation de la fonction de score nativement implémentée dans Surflex Dock n'améliore que très peu la qualité du filtre par rapport au logiciel Dock (voir courbes de ROC sur la figure 6.19 page suivante et AUCs correspondantes sur le tableau 6.3 page 143).

Du fait que les poses générées lors d'un calcul de *docking* sont différentes selon les logiciels utilisés, une nouvelle analyse de contacts peut être réalisée sur les poses générées par Surflex Dock. Sur la figure 6.20 page 149, l'analyse des contacts montre clairement une amélioration de la sensibilité et de la spécificité du criblage ainsi effectué. L'AUC passe de 0,58 pour le filtre de score, à 0,87 lorsque le logiciel AuPosSOM est utilisé (tableau 6.3 page 143).

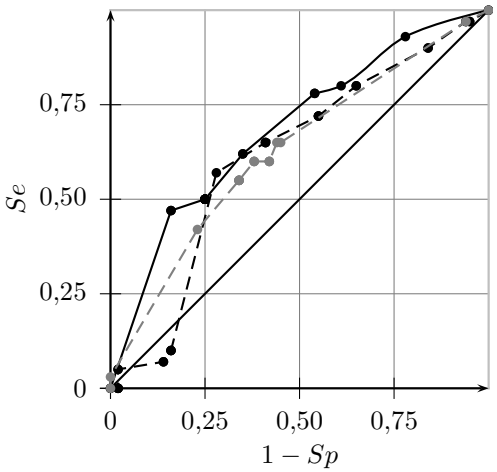


FIGURE 6.18 – Comparaison des algorithmes SOM (ligne pleine), *k-means* avec $k = 10$ (ligne pointillée) et *k-means* avec $k = 20$ (ligne grise pointillée).

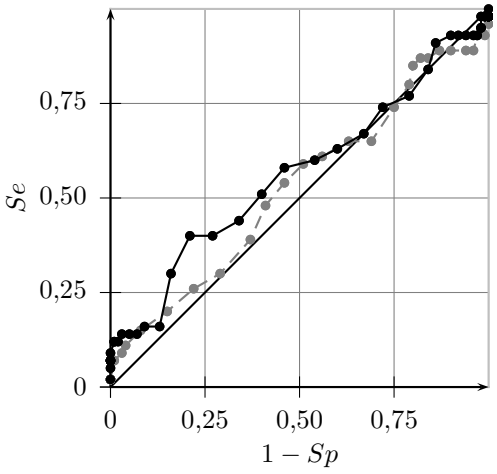


FIGURE 6.19 – Comparaison du logiciel Surflex Dock (courbe noire) et Dock (courbe grise) en utilisant une sélection par score avec les fonctions implémentées dans chacun de ces deux logiciels.

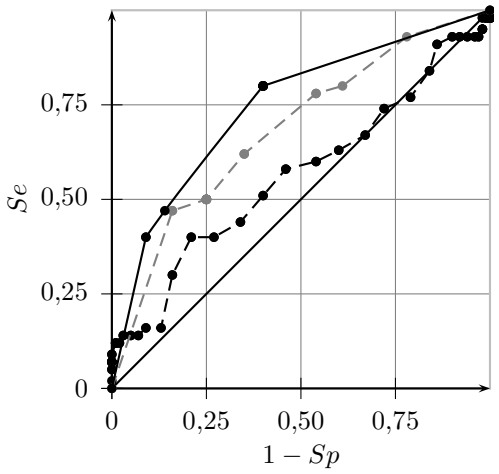


FIGURE 6.20 – Comparaison d'un criblage utilisant le score Surflex Dock (ligne pointillée noire), et l'analyse de contact des poses générées par Surflex Dock (ligne continue). Est rappelée en ligne pointillée grise la courbe obtenue lors de l'analyse des contacts générées par Dock.

La comparaison de l'analyse des contacts par carte de Kohonen effectuée sur les poses générées par Surflex Dock avec celles générées par Dock montre une nette amélioration du processus de criblage lorsque Surflex Dock est utilisé. L'AUC passe alors de 0,76 pour Dock à 0,87 pour Surflex Dock. Ainsi le positionnement des molécules est meilleur avec le logiciel Surflex Dock par rapport à la qualité des poses générées par Dock. Cependant cette constatation n'est pas visible sur la qualité du *scoring* effectuée par Surflex Dock par rapport à Dock (voir figure 6.19 page précédente). L'ensemble de ces résultats montre l'importance de la qualité du *docking* dans le protocole de sélection par homologie de contacts.

L'utilisation d'autres fonctions de score est souvent employée comme étape de *post-processing* (voir partie 3.1.6 page 50). Afin de tester les performances de ces fonctions de score par rapport aux résultats obtenus avec le logiciel AuPosSOM, nous avons testé quatre fonctions de score, à savoir : G_Score, PMF_Score, D_Score et ChemScore. Pour la cible considérée ces fonctions de score donnent de meilleurs résultats que la fonction interne à Surflex Dock (voir figure 6.21 page suivante et tableau 6.3 page 143).

La comparaison de ces différentes fonctions de score avec l'analyse de contact réalisée par AuPosSOM montre que celle ci permet d'obtenir le même point de plus faible coefficient γ que la meilleure fonction de score de *post-processing*, à savoir G_Score, pour la cible considérée. Cependant, du fait

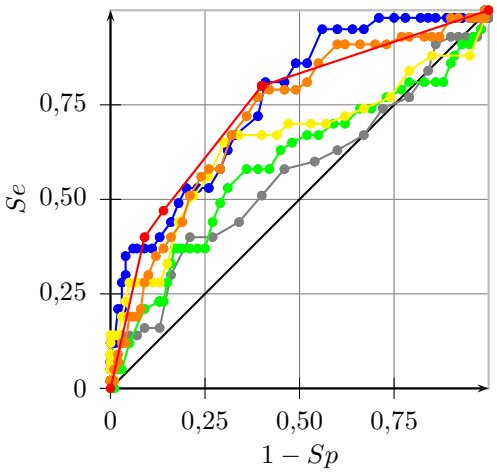


FIGURE 6.21 – Comparaison des fonctions de score Surflex Dock (gris), G_Score (bleu), PMF_Score (vert), D_Score (jaune) et ChemScore (orange). En rouge est représenté le résultat de l'analyse des contacts avec le logiciel AuPosSOM.

du mode de tracé de la courbe de ROC il est impossible d'obtenir autant de points que pour la courbe de ROC obtenue avec le score. Malgré ces considérations techniques, les points de la courbe de ROC obtenus avec AuPosSOM montrent toujours une plus grande spécificité à sensibilité égale.

Chapitre 7

Recherche de nouveaux anti-intégrase du VIH-1

7.1 Criblage virtuel sur l'intégrase du VIH-1

Le but de ce criblage est de trouver des inhibiteurs de l'intégrase. Il doit aussi être capable de distinguer les inhibiteurs catalytiques des inhibiteurs non-catalytiques visant potentiellement l'inhibition de l'interaction entre l'intégrase et le LEDGF. Ne connaissant pas le site de fixation de ces derniers il est nécessaire de réaliser un docking sur l'ensemble de la surface moléculaire de l'intégrase. Cette considération complique le protocole de criblage virtuel et nécessite le développement d'une stratégie originale. Nous allons envisager un protocole de *docking* à l'aveugle ou *blind docking*, englobant la totalité de la surface moléculaire.

7.1.1 Génération d'un modèle pour le récepteur

La structure pdb 2b4j [23] a été choisie comme cible pour effectuer le criblage virtuel. Cette structure cristallographique donne les coordonnées atomiques du dimère constitué de deux domaines catalytiques de l'intégrase (CCD) en interaction avec deux domaines de liaison de l'intégrase (IBD) du LEDGF. Cette structure est représentée sur la figure 7.1 page suivante.

Afin de docker les molécules sur l'ensemble de la surface moléculaire, y compris sur la zone d'interface avec le LEDGF, les coordonnées atomiques des deux IBD du LEDGF ont été retirées.

L'intégrase du VIH-1 catalyse deux réactions, à savoir, le transfert de brin et le traitement de l'extrémité 3' de l'ADN viral. Il a été démontré que les deux réactions catalysées par l'intégrase nécessitent la présence de métaux

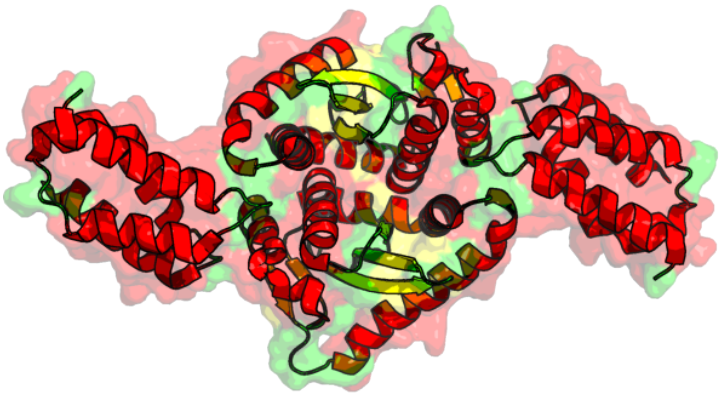


FIGURE 7.1 – Structure cristallographique (code pdb : 2b4j) du complexe intégrase (CCD)·LEDGF (IBD). Le dimère d'intégrase CCD_2 est représenté au centre et est entouré de deux IBD du LEDGF. Les hélices α sont représentées en rouge, les brins β en jaune et les boucles en vert.

bivalents comme le Mn^{2+} ou le Mg^{2+}, ce dernier étant celui présent physiologiquement [110]. Malgré le lancement du raltégravir en clinique la structure des inhibiteurs en interaction avec l'intégrase n'est pas encore connue [97].

La structure du complexe présentée dans la figure 7.1 page ci-contre a été obtenue en l'absence de métaux bivalents au niveau du site actif de l'intégrase. Afin de trouver de potentiels inhibiteurs catalytiques de cette molécule il est nécessaire d'avoir un modèle où les deux ions magnésium sont présents au niveau du site catalytique. Des travaux réalisés par Andrea Savarino [155] propose un modèle théorique du domaine catalytique de l'intégrase en interaction avec deux ions magnésium. Cette structure vient de la structure cristallographique 1bl3 [109] qui donne les coordonnées atomiques du domaine catalytique de l'intégrase en interaction avec un magnésium. La structure de l'intégrase du virus du sarcome aviaire, structure 1vsh [15], a été résolue en présence de deux cations bivalent au niveau du site actif. La superposition de la structure 1bl3 avec la structure 1vsh a permis d'obtenir un modèle pour le site actif de l'intégrase du VIH-1 en présence de deux ions magnésium. Les différentes structures, 1bl3 et 1vsh, ainsi que le modèle obtenu sont présentés en figure 7.2 page suivante.

Cependant le modèle obtenu par Savarino [155] n'est pas issu de la structure de l'intégrase en interaction avec le LEDGF, donc présente des différences structurales au niveau de l'interface entre l'intégrase et le LEDGF. Un modèle du complexe intégrase·LEDGF contenant deux ions magnésium au niveau du site catalytique a été construit. Le logiciel Modeller [43] a été utilisé afin d'obtenir une protéine modèle conservant la structure du site catalytique du modèle de Savarino et la zone d'interaction avec le LEDGF intactes. La figure 7.3a page 155 donne les fragments des deux structures de départ qui ont été utilisés afin de produire le modèle du complexe intégrase·LEDGF contenant deux ions magnésium. A partir de ces deux structures le programme Modeller a permis de générer une structure modèle du complexe intégrase·LEDGF contenant deux ions magnésium (figure 7.3b page 155).

La recherche de sites de fixation sur le dimère CCD_2 a ensuite été réalisée par le logiciel SiteHound (voir paragraphe 7.1.2 page 158). Dix-huit sites potentiels de fixation (*clusters*) ont été trouvés, et les cinq de plus basses énergies ont été conservés pour les études par *docking* (voir figure 7.4 page 156 et tableau 7.1 page suivante).

Le site 1, de plus basse énergie, est au niveau du site catalytique, ainsi que le site 4. Ces deux sites entourent les deux ions magnésium, ceux ci n'étant pas détecté par la sonde carbone EasyMIF utilisée. Ces deux sites stabilisent la partie hydrophobe des composés se liant au site catalytique [164].

Le site 2, d'énergie et de volume comparables au site catalytique (*cluster* 1), se trouve au sein d'une cavité à l'interface entre les deux monomères du

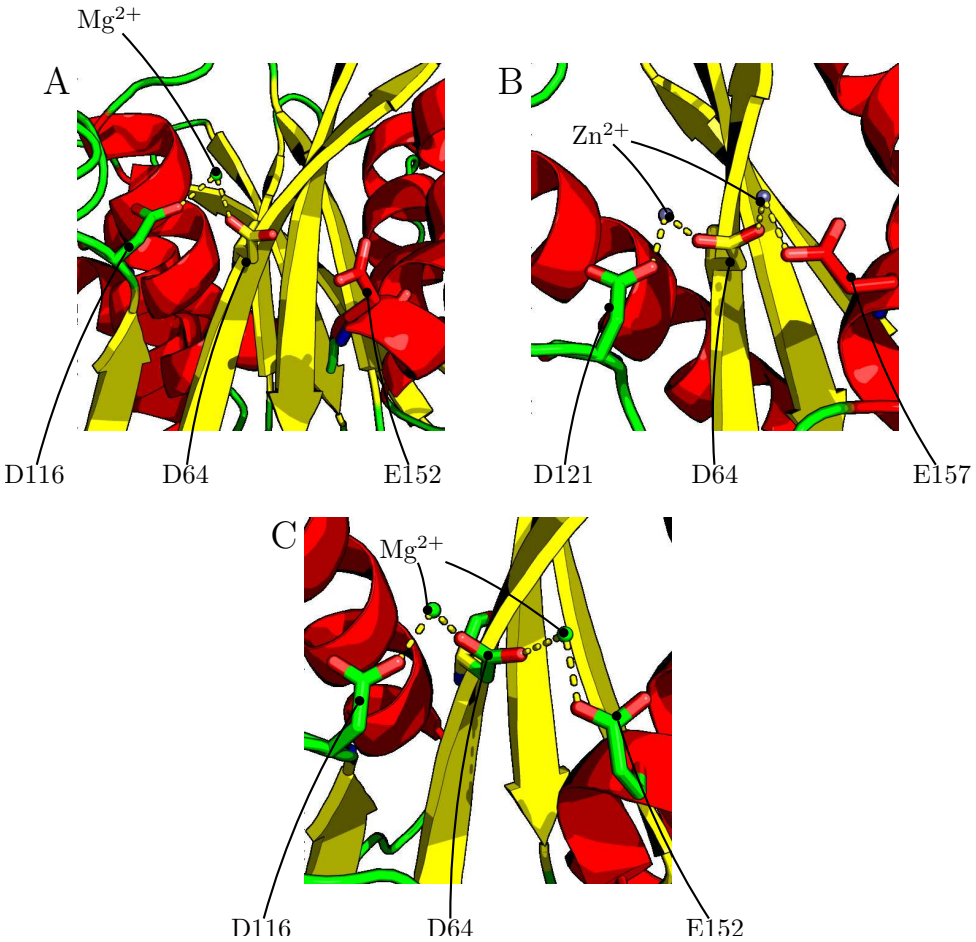


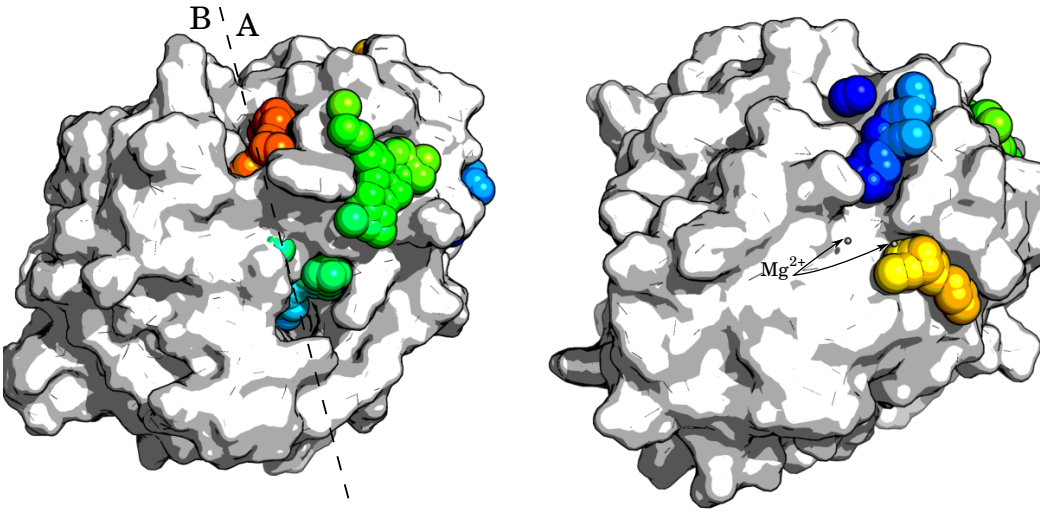
FIGURE 7.2 – Vue du site catalytique de l'intégrase du VIH-1, structure 1bl3 (**A**), de l'intégrase du virus du sarcome aviaire, structure 1vsh (**B**) et du modèle d'intégrase contenant deux ions magnésium [155] (**C**) résultant de l'étude par superposition des structures présentées en A et B.

n° *cluster*	Énergie ($kcal \cdot mol^{-1}$)	Volume ($Å^3$)	Position
1	-470,791	41	Catalytique
2	-466,102	42	Dimère
3	-399,180	39	W131
4	-217,053	19	Catalytique
5	-199,788	20	Poche LEDGF

TABLE 7.1 – *Clusters* identifiés par le logiciel SiteHound sur le dimère CCD_2.

(a) (b)

FIGURE 7.3 – Modélisation d'une structure chimère (b) entre la structure 2b4j du complexe IN·LEDGF (en vert sur la figure a) et le modèle de l'intégrase contenant les deux ions magnésium (en rouge sur la figure a). Le complexe CCD_2·$LEDGF_2$ obtenu est présenté en b. Les ions magnésium sont mis en évidence par les deux sphères vertes.



(a) *Clusters* 2, 3 et 5 (b) *Clusters* 1 et 4

FIGURE 7.4 – Sites de fixation prédits par le logiciel SiteHound sur le modèle CCD_2. Les couleurs froides indiquent les basses énergies (sites de fixation stables), les couleurs chaudes les plus hautes énergies. Les *clusters* sont classés par énergie croissante, du plus stable au moins stable. La ligne pointillée indique la frontière entre les deux monomères A et B.

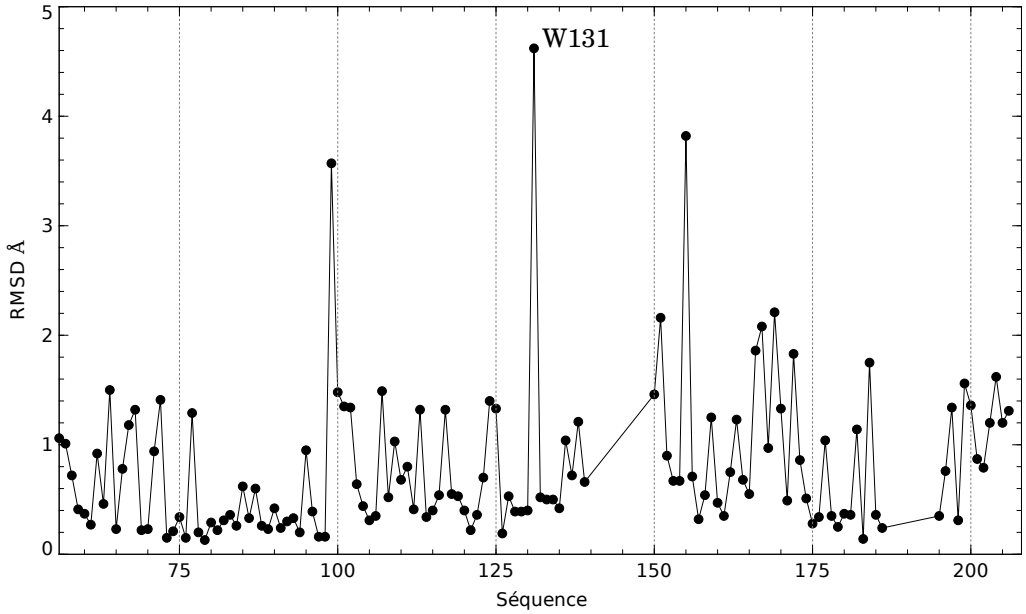


FIGURE 7.5 – RMSD local entre la structure cristallographique du dimère CCD_2 (code pdb : 1bl3) et la structure du dimère CCD_2 extraite de la structure cristallographique du complexe CCD_2·IBD_2 (code pdb : 2b4j).

dimère CCD_2. L'interface entre les deux domaines CCD de l'intégrase font l'objet de nombreuses recherches pour le développement d'inhibiteurs [159, 180]. Ce site est proche de l'hélice α1 qui est importante à la dimérisation de l'intégrase. L'interaction homotope entre les hélices α1 des deux dimères est importante à la dimérisation. Le peptide ^{97}TAYFLLKLAGRW^{108} issu de la séquence de l'hélice α1 empêche la dimérisation de l'intégrase [112].

Le site 3 est proche du tryptophane 131 (W131). Cet acide aminé hydrophobe est important dans la fixation du LEDGF. De plus le peptide ^{128}AACWWAGIK^{136} a été identifié comme site de fixation d'inhibiteurs de l'intégrase du VIH-1 [2]. De manière intéressante, le tryptophane 131 est très exposé au solvant dans la structure cristallographique de l'intégrase non complexée au LEDGF (code pdb : 1bl3 [109]), tandis qu'il est impliqué dans l'interaction avec l'IBD du LEDGF dans la structure du complexe CCD·IBD (code pdb : 2b4j – voir figure 2.6 page 29). Le calcul du RMSD local entre ces deux structures montre que sa position est celle la plus influencée par la fixation de l'IBD du LEDGF (voir figure 7.5).

Le site 5 reçoit la boucle 1 de liaison de l'intégrase de l'IBD du LEDGF (voir figure 7.1 page 152). Le peptide ^{361}NSLKIDNLDV^{370}, extrait de la

boucle de l'IBD du LEDGF, inhibe l'intégrase *in-vitro* et *in-cellulo* [65].

Ainsi sur les dix-huit sites trouvés par SiteHound les cinq premiers sites – de plus basses énergies – sont tous reliés à des données bibliographiques concernant l'inhibition de l'intégrase.

Cinq boîtes de *docking* ont été créé afin d'effectuer les expériences de *docking*. Le centre des boîtes est confondu avec le barycentre de chacun des *clusters* et le volume de la boîte est ajusté au volume du *cluster* (voir tableau 7.1 page 154).

7.1.2 *Blind docking*

Un premier *blind docking* a été réalisé sur la surface protéique du modèle CCD_2 généré comme décrit dans le paragraphe 7.1.1 page 151. Afin de réduire la surface à explorer et d'éviter une divergence de l'ensemble des poses de *docking* la grille de *docking* n'englobe qu'un monomère et la zone d'interface entre les deux monomères constituant le dimère. Chaque *docking* permet de générer vingt poses.

Un criblage d'une chimiothèque privée fournie par la société CellVir SAS a été réalisé, en utilisant pour le *docking* le logiciel AutoDock 4.0. Le nombre de composés dockés est de 4 260. L'analyse des empreintes de contact par le logiciel AuPosSOM a permis de générer l'arbre présenté en figure 7.6 page ci-contre. Les composés se liant potentiellement au niveau du site catalytique sont facilement identifiables en analysant les empreintes de contact des différentes branches. La branche indiquée par la légende "composés catalytiques" présente une empreinte caractéristique comme indiquée en figure 7.8d page 161. Sur cette empreinte les trois acides aminés de la triade catalytique – D64, D121 et E157 – sont contactés fréquemment.

La sélection de la branche contenant des composés ayant potentiellement une activité anti-intégrase a été réalisée à l'aide d'une molécule (notée CVR224) détectée par CellVir SAS comme ayant une activité anti-intégrase non catalytique. Cette molécule a été dockée comme l'ensemble des molécules de la chimiothèque et se retrouve dans la branche indiquée par la légende "composés non catalytiques" sur la figure 7.6 page ci-contre.

Les 51 molécules constitutives des deux branches mises en évidence sur la figure 7.6 page suivante ont ensuite étaient extraites de la chimiothèque et les vecteurs correspondant ont servi à l'entraînement d'une nouvelle carte de Kohonen de dimension 5×4. Ce nouvel apprentissage permet de classer plus précisément ces molécules. et de générer un arbre plus lisible. Le résultat obtenu est présenté sur la figure 7.7 page 160. Cet arbre non enraciné montre quatre branches distinctes qui correspondent respectivement au site proche du tryptophane 131 (W131), au site à l'interface des deux

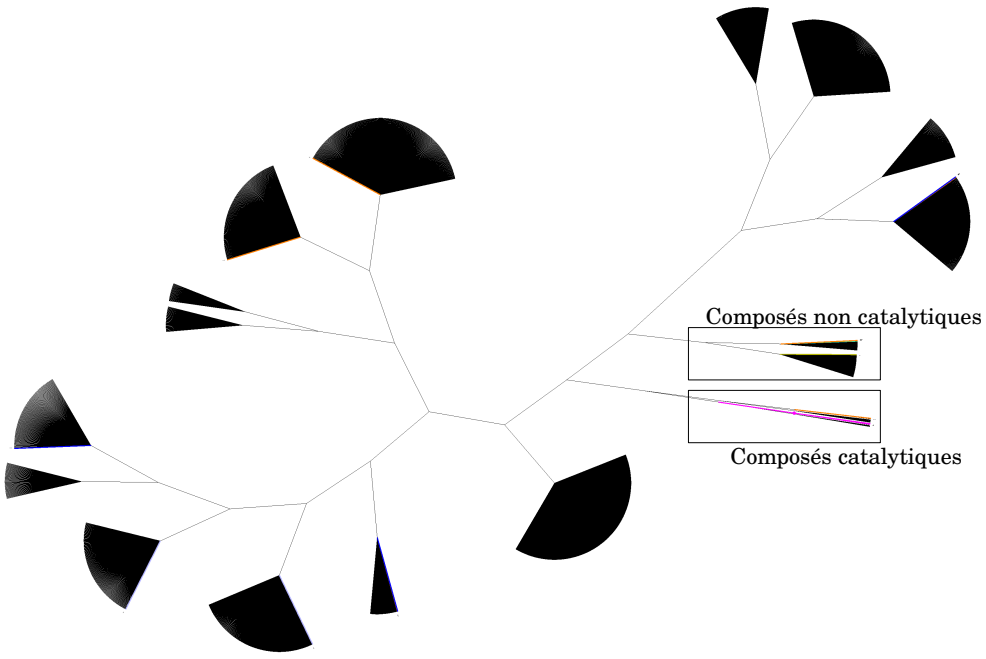


FIGURE 7.6 – Regroupement hiérarchique des molécules après analyse des contacts des résultats d'un criblage virtuel de 4 260 composés sur le dimère CCD_2 de l'intégrase du VIH-1.

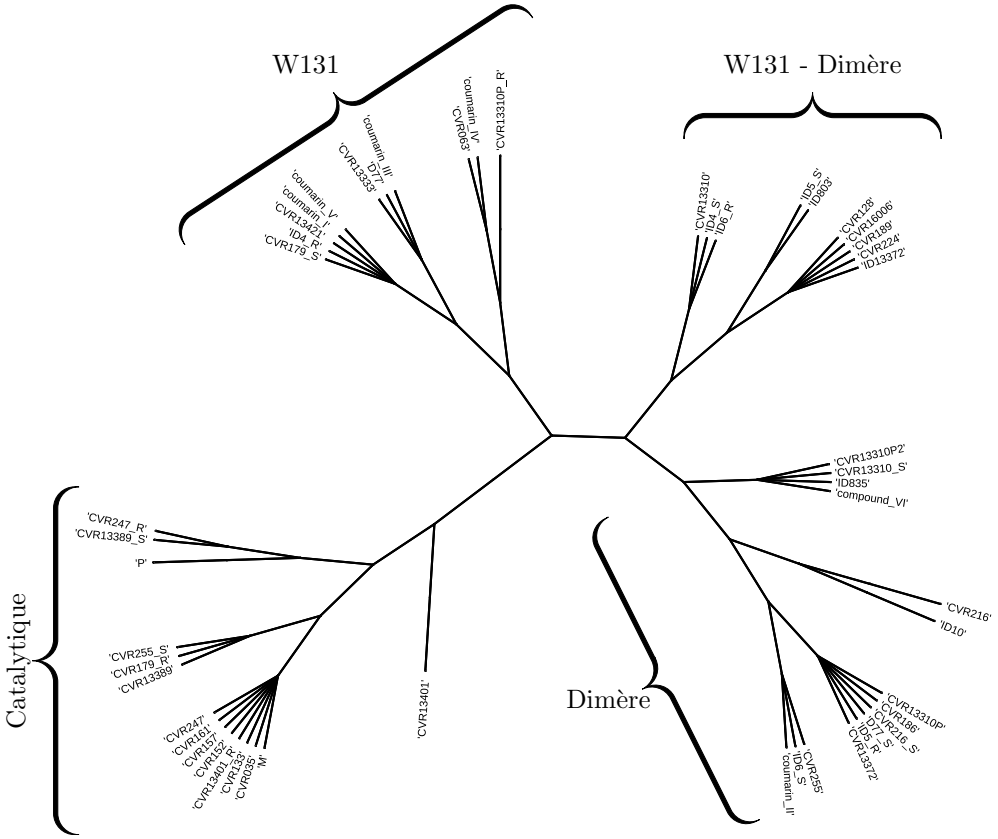
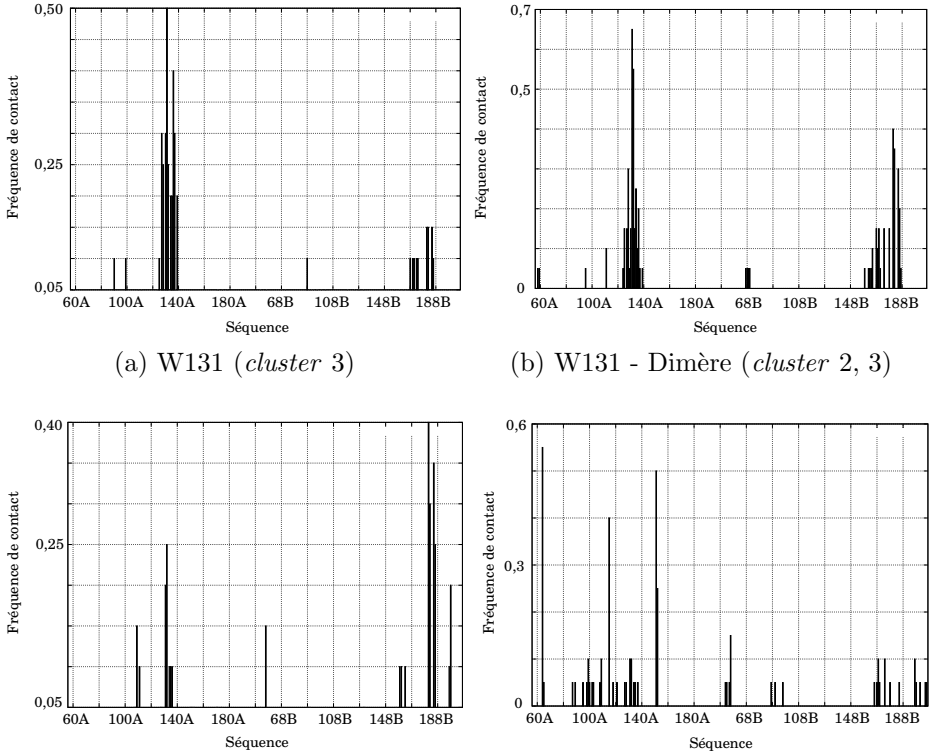


FIGURE 7.7 – Arbre résultant de l'analyse AuPosSOM du *blind docking* effectué sur le dimère CCD_2 de l'intégrase du VIH-1.



(a) W131 (*cluster* 3) (b) W131 - Dimère (*cluster* 2, 3)

(c) Dimère (*cluster* 2) (d) Catalytique (*cluster* 1, 4)

FIGURE 7.8 – Empreintes caractéristiques des différents sites identifiés par le logiciel AuPosSOM. La séquence des acides aminés est numéroté selon la position dans la chaîne et l'identifiant de la chaîne (A ou B). Le numéro des *clusters* SiteHound (voir tableau 7.1 page 154) correspondant à l'empreinte est indiqué.

monomères (dimère) au site catalytique, et à un site mixte touchant à la fois l'interface de dimérisation et le tryptophane 131 (W131 - Dimère). De manière intéressante ces quatre sites avaient aussi été trouvé par l'analyse des MIFs par le logiciel SiteHound. Les sites trouvés par l'analyse des contacts sont à mettre en relation avec les *clusters* 1, 2 et 3 trouvés par le logiciel SiteHound. Ces *clusters* sont les 3 sites de plus basse énergie (voir tableau 7.1 page 154) trouvés par le programme SiteHound. Les graphiques présentés sur la figure 7.8 sont les empreintes caractéristiques des sites identifiés.

L'empreinte du site W131 (figure 7.8a) montre que la majorité des poses contacte la région du tryptophane 131 de la chaîne A. Une faible proportion des poses, inférieure à 15% contacte aussi le monomère B.

L'empreinte du site "W131 - dimère" (figure 7.8b page précédente) est proche de l'empreinte du site W131. Cependant la zone de la chaîne B contactée peu fréquemment pour l'empreinte correspondant au site W131 est contactée plus souvent pour ce site : 40% des poses contacte la chaîne B.

L'empreinte caractéristique du site dimère (figure 7.8c page précédente) montre qu'en moyenne plus de 30% des poses contacte la chaîne B et 25% la chaîne A.

L'empreinte du site catalytique (figure 7.8d page précédente) présente trois points de contact très fréquent. Ces trois points sont les trois acide-aminés constitutifs de la triade catalytique de l'intégrase : D64, D121 et E157.

Les distributions des contacts des sites W131, W131 - dimère et dimère présentent des analogies. La distribution des contacts caractéristiques du site catalytique présente moins d'homologie avec le reste des sites. Ceci est visible directement par l'éloignement de la branche des catalytiques sur le graphique en arbre présenté en figure 7.7 page 160.

La figure 7.9 page ci-contre donne la répartition de la fréquence des contacts sur la structure tri-dimensionnelle du dimère CCD_2 pour les différents sites identifiés.

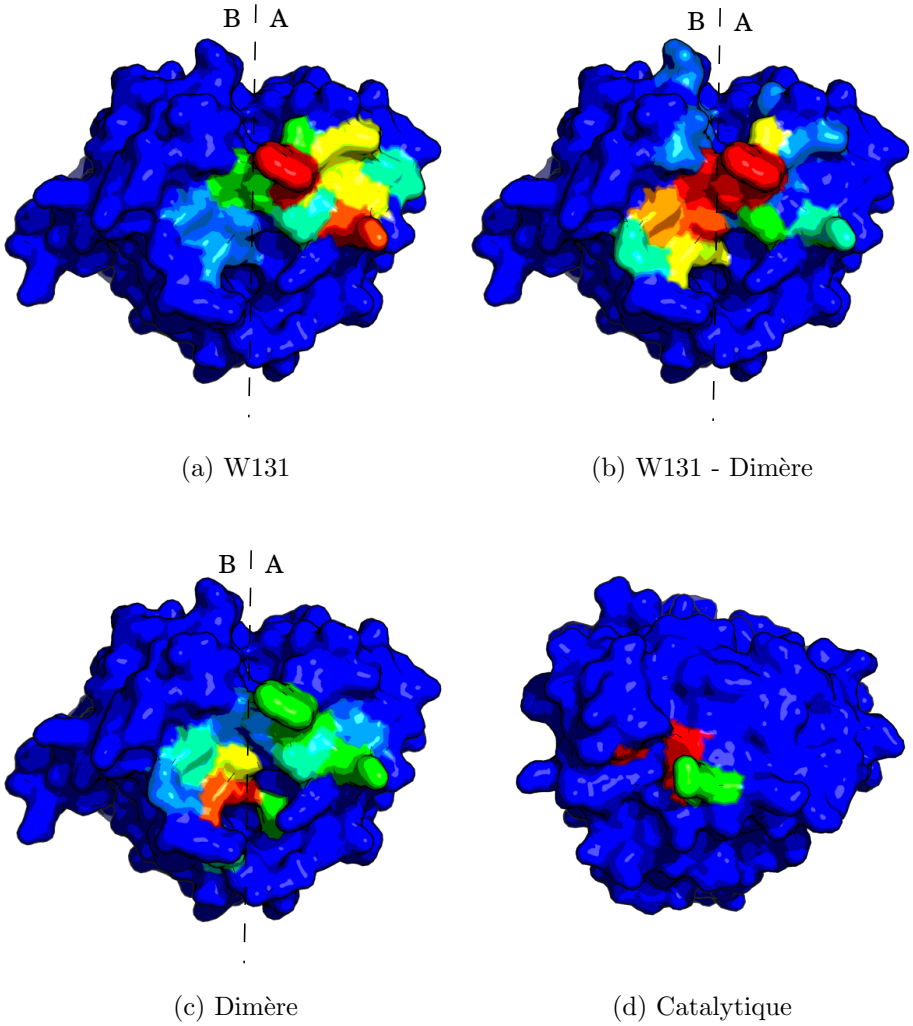
7.2 Étude par RMN des interactions ligands-intégrase du VIH-1

Les expériences RMN employées permettent de tester l'interaction avec l'intégrase des molécules prédites comme active par les calculs réalisés lors du criblage virtuel. Les séquences STD et WaterLOGSY permettent de mettre en évidence une interaction mais donnent aussi accès à des informations structurales concernant les distances entre protons du ligand et de la protéine. La combinaison des données expérimentales RMN avec les données théoriques de modélisation moléculaire devraient permettre de valider et d'affiner les modèles.

7.2.1 Mise en évidence et études de ligands catalytiques de l'intégrase

Étude du composé GS9137 comme ligand catalytique modèle

Le composé GS9137, dérivé de la famille des antibiotiques quinolones, est un composé qui inhibe la réplication du VIH-1. Cette molécule est capable de lier les cations magnésiums. Elle inhibe l'étape de transfert de brin du cycle



(a) W131

(b) W131 - Dimère

(c) Dimère

(d) Catalytique

FIGURE 7.9 – Empreintes caractéristiques des différents sites identifiés par le logiciel AuPosSOM représentées sur la structure du dimère CCD_2. Le code couleur indique la fréquence de contact, les couleurs chaudes représentant les fréquences élevées, les couleurs froides les basses fréquences. Les lignes pointillées indiquent les frontières entre les deux monomères A et B.

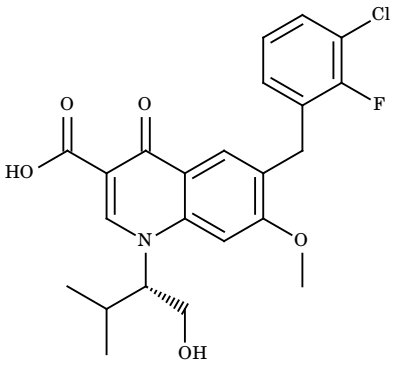


FIGURE 7.10 – Formule développée plane du GS9137

catalytique de l'intégrase du VIH-1 [154]. La formule développée plane du GS9137 est donnée en figure 7.10. L'interaction du GS9137 avec le complexe intégrase·LEDGF a été étudiée par RMN. Le spectre WaterLOGSY résultant est donné en figure 7.11 page ci-contre. Le ratio concentration en ligand sur concentration en protéine ($[L]/[P]$) est de 100. L'expérience WaterLOGSY met en évidence l'interaction entre le complexe intégrase·LEDGF et le composé GS9137. Les signaux d'une impureté à 8,4 ppm, du DMSO à 2,7 ppm et du glycérol à 3,6 ppm sont d'intensités négatives, signifiant que les NOEs correspondant sont négatifs. Ces molécules ne se lient donc pas au complexe.

Le mode de liaison du composé GS9137 sur l'intégrase du VIH-1 reste inconnu. Cependant la structure cristallographique de l'intégrase entière du PFV, en interaction avec le GS9137, a été résolue très récemment (code PDB : 3l2u) [62].

Afin d'obtenir un modèle de liaison du GS9137 en interaction avec l'intégrase du VIH-1, ce composé a été docké sur la structure de l'intégrase obtenue selon le protocole présenté dans le paragraphe 7.1.1 page 151. Le logiciel Dock 6.3 a été utilisé pour réaliser ce *docking*. La figure 7.12a page 166 montre la conformation du GS9137, de plus basse énergie, prédite au sein du site actif de l'intégrase du VIH-1 et la figure 7.12b page 166 montre en détail la liaison aux cations magnésium dans cette structure de *docking*. La comparaison du mode de liaison du GS9137 aux cations magnésium du site actif de l'intégrase du VIH-1 avec les données expérimentales obtenues par diffraction des rayons X sur le complexe entre l'intégrase du PFV et cette même molécule (figure 7.12c page 166) montre des conformations comparables.

L'analyse du spectre WaterLOGSY en comparaison avec le spectre RMN 1D permet de déterminer les protons les plus en interaction avec la protéine. Une normalisation à 1 est appliquée aux protons les plus impliqués dans

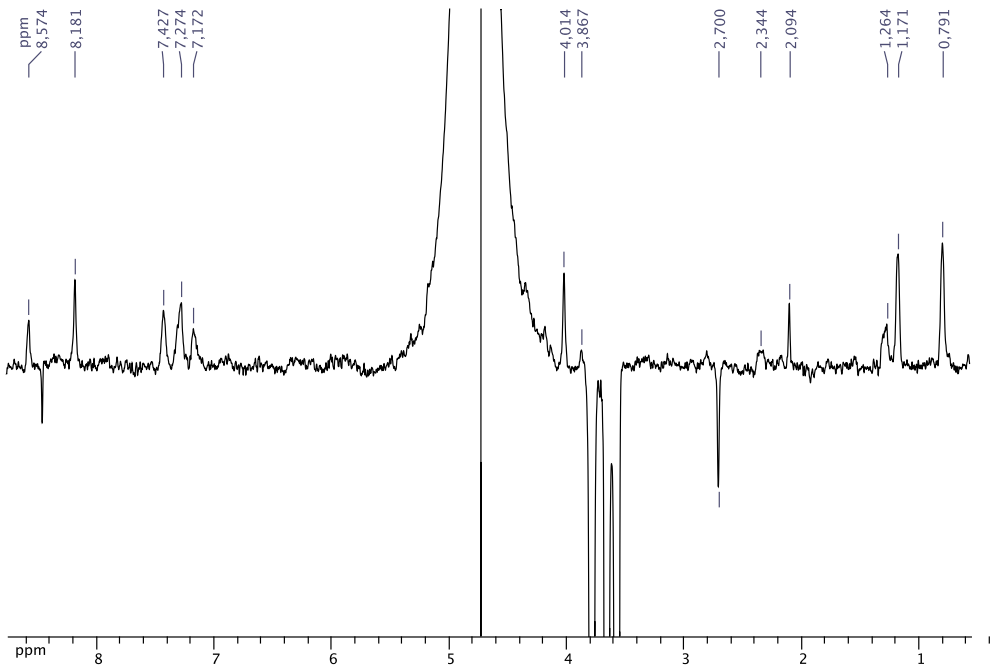
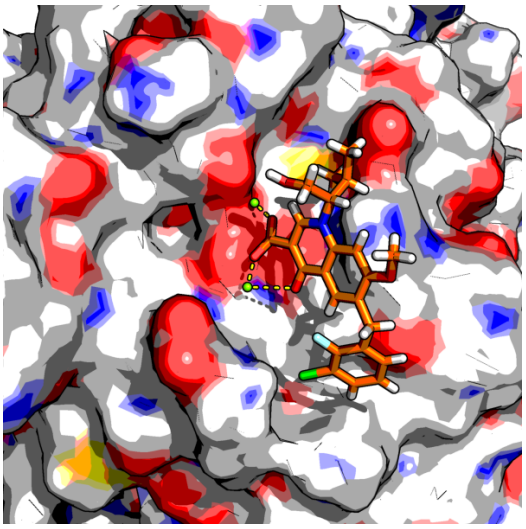
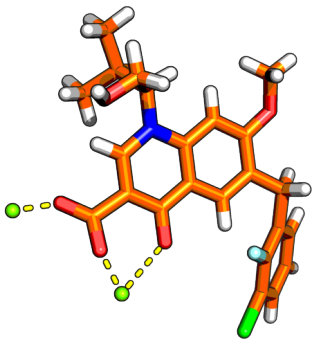


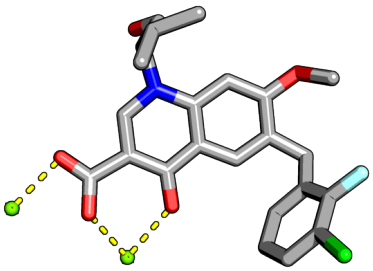
FIGURE 7.11 – Spectre WaterLOGSY-*excitation-sculpting* du GS9137 en présence du complexe intégrase·LEDGF.



(a) *Docking* du GS9137 sur l'intégrase du VIH-1

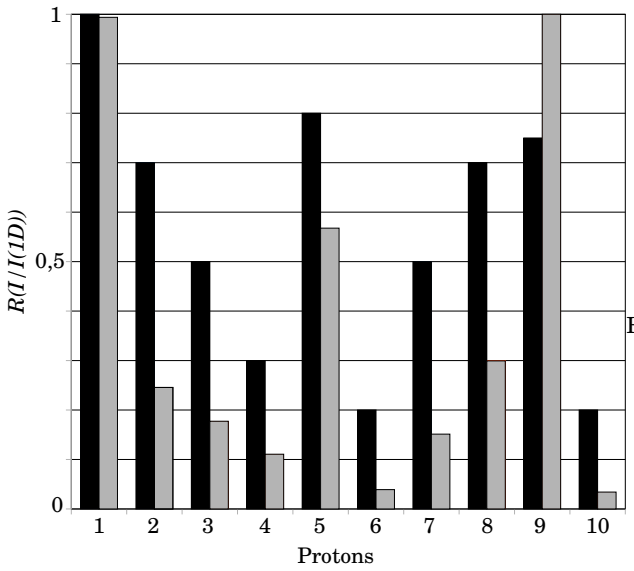


(b) Détail de la structure de *docking* montrant la liaison aux cations magnésium du site actif

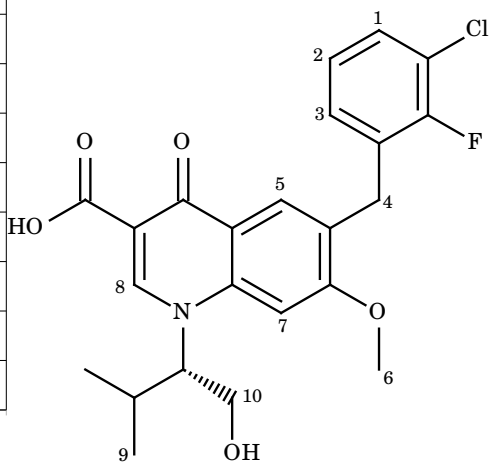


(c) Détail de la structure 3l2u montrant la liaison aux cations magnésium du site actif pour l'intégrase du PFV

FIGURE 7.12 – Modèle de liaison du GS9137 au site actif de l'intégrase du VIH-1 (a & b). Comparaison aux données cristallographiques obtenues pour le complexe entre l'intégrase du PFV et le GS9137 (c).



(a) *Epitope mapping* WaterLOGSY du GS9137 – en noir –, comparé avec la structure de *docking* (figure 7.12a page précédente) – en gris –



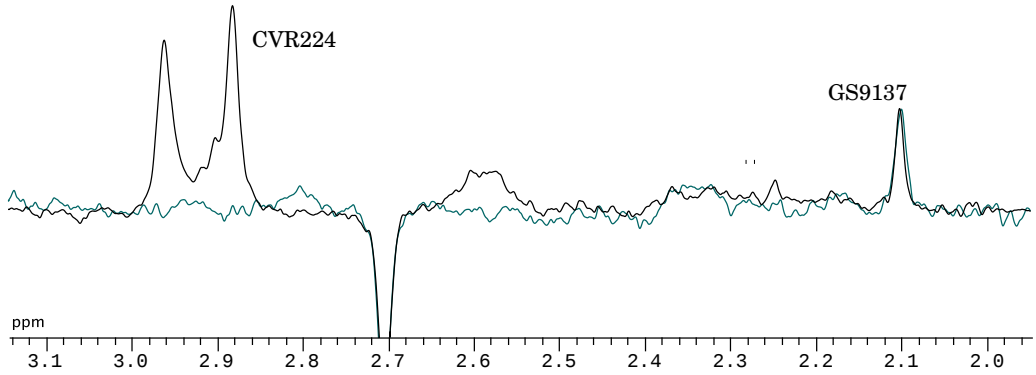
(b) Formule développée plane du GS9137

FIGURE 7.13

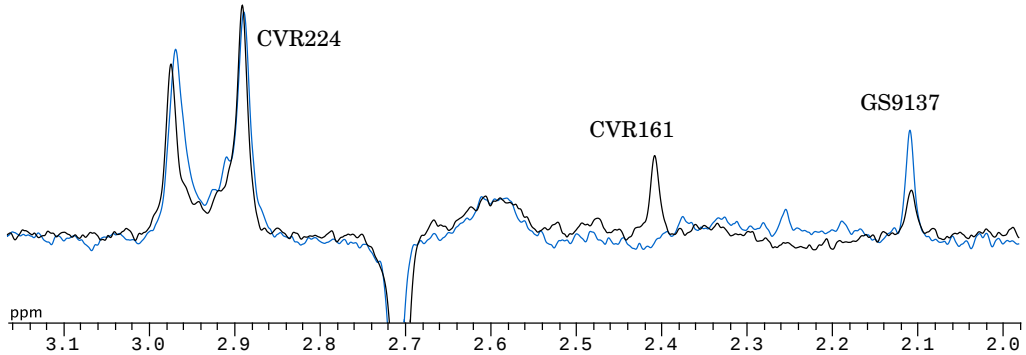
l'interaction. La figure 7.13 donne les résultats de l'*epitope mapping* pour le GS9137 en interaction avec le complexe intégrase·LEDGF. La pose de *docking* considéré corrèle bien avec l'*epitope mapping* déterminé expérimentalement. Le coefficient de corrélation est de 0,83. Les protons 1, 2, 5, 8 et 9 sont fortement impliqués dans l'interaction avec l'intégrase. Le groupement isopropyl est impliqué dans des interactions de van de Waals avec la protéine. L'*epitope mapping* montre clairement une orientation du motif quinolone du coté de la protéine, qui s'explique par la chélation des cations magnesium par celui ci. La groupement halobenzyle se positionne au sein d'une poche et stabilise ainsi la structure. Ces résultats sont en accord avec la structure cristallographique obtenue avec l'intégrase du PFV.

Découverte d'un nouveau ligand du site catalytique de l'intégrase

L'arbre présenté en figure 7.7 page 160 a permis de sélectionner un ensemble de molécules pouvant potentiellement se fixer au niveau du site catalytique de l'intégrase. L'étude de ces molécules par RMN a permis d'identifier la molécule CVR161 comme ligand du site catalytique de l'intégrase. Le site de fixation du CVR161 a été mis en évidence par l'utilisation de deux composés utilisés comme rapporteurs d'un site non catalytique et du site catalytique



(a) Spectres WaterLOGSY du GS9137 seul (en vert) et du mélange CVR224/GS9137 (n/n : 0,5/1) (en noir).



(b) Spectres WaterLOGSY du mélange CVR224/GS9137 (en bleu) et du mélange CVR161/CVR224/GS9137 ($n/n/n$: 0,6/0,5/1) (en noir).

FIGURE 7.14 – Région aliphatique des spectres WaterLOGSY du GS9137, du CVR224 et du CVR161 en interaction avec le complexe intégrase·LEDGF.

de l'intégrase. Ces composés sont respectivement le CVR224 et le GS9137.

La figure 7.14a, montre que le CVR224 n'est pas en compétition avec le GS9137. En effet l'ajout de CVR224 n'entraîne pas une diminution du signal WaterLOGSY du GS9137. Cependant l'ajout de CVR161 entraîne une diminution du signal du GS9137 mais n'influence pas le signal du CVR224 (figure 7.14b). Ce résultat met en évidence une compétition entre le GS9137 et le CVR161. Ces deux composés se lient donc au même site, donc le composé CVR161 est bien un ligand du site catalytique comme prédit par le modèle développé (voir figure 7.7 page 160). Des tests réalisés par la société CellVir SAS ont corroboré ces résultats en montrant que le composé CVR161 possède des propriétés inhibitrices de l'activité catalytique de transfert de brin de l'intégrase.

La structure de *docking* obtenue pour le composé CVR161 est présentée sur la figure 7.15 page suivante. Comme pour le composé GS9137, le CVR161 chélate les deux ions magnésium du site actif de l'intégrase du VIH-1.

L'*epitope mapping* à partir du spectre WaterLOGSY a été réalisé pour le CVR161 (voir figure 7.16 page 171). La comparaison avec la pose de *docking* présenté en figure 7.15a page suivante montre une bonne corrélation des données expérimentales et théoriques : le coefficient de corrélation est de 0,74. L'*epitope mapping* montre une faible implication du groupe méthyle de la molécule ce qui est dû à l'orientation du CVR161 par la chélation des deux cations magnésium.

7.2.2 Mise en évidence et études de ligands non catalytiques de l'intégrase

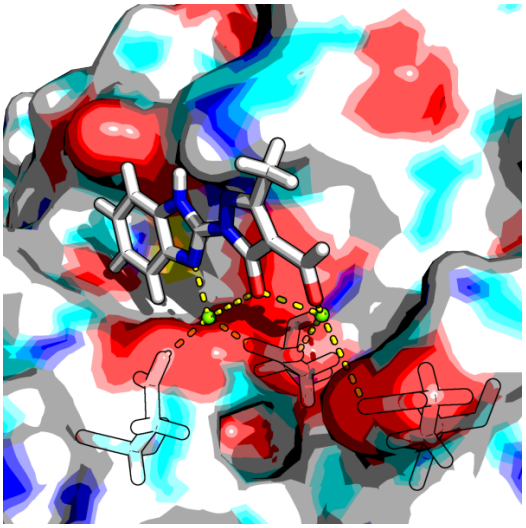
Le composé CVR224 est un ligand du complexe intégrase·LEDGF, il possède des propriétés anti-virales sur le VIH-1 mais ne se lie pas au niveau du site catalytique, comme nous l'avons montré par les expériences de compétition avec le composé du site catalytique GS9137 (voir figure 7.14a page précédente).

Afin de montrer que ce ligand se lie au niveau de l'intégrase, il est nécessaire de réaliser des expériences RMN d'interaction en n'utilisant non pas le complexe intégrase·LEDGF mais l'intégrase seule. Ces expériences sont plus délicates à réaliser du fait de l'instabilité plus grande de l'intégrase lorsqu'elle n'est pas complexée au LEDGF. De plus du fait que l'efficacité du transfert d'aimantation par NOE augmente avec la masse moléculaire du récepteur (voir partie 5.3 page 99), l'utilisation de l'intégrase seule ($MM =$ 140 kDa, pour l'homodimère IN_2 portant l'étiquette de purification MBP [1] [50]) est moins propice aux expériences de transfert d'aimantation que lorsque le complexe intégrase·LEDGF est utilisé ($MM = 250$ kDa pour le complexe IN_2·$LEDGF_2$ avec les étiquettes de purification GST et His_6 respectivement – voir partie 5.5.2 page 108).

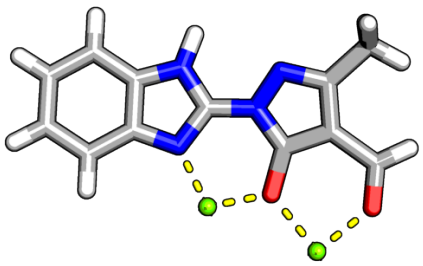
Des expériences STD, WaterLOGSY et RNP ont été réalisées sur le modèle intégrase seule, non complexé au LEDGF (voir figure 7.17 page 172). L'ensemble des expériences RMN réalisées montre que le CVR224 interagit avec l'intégrase du VIH-1. Les signaux des impuretés ne correspondant pas aux signaux du CVR224 sont bien éliminées dans les expériences WaterLOGSY, STD et RNP.

L'*epitope mapping* du CVR224 a été réalisé en considérant l'interaction avec le complexe intégrase·LEDGF ainsi qu'avec l'intégrase seule. Les

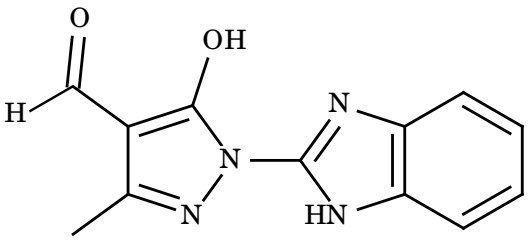
1. *Maltose Binding Protein*



(a) *Docking* du CVR161 sur l'intégrase du VIH-1

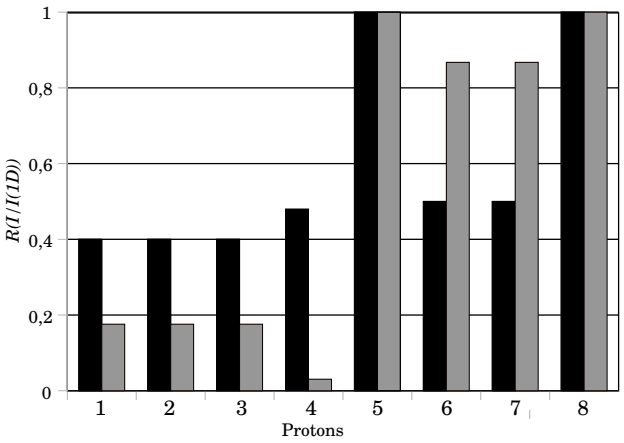


(b) Détail de la structure de *docking* montrant la liaison aux cations magnésium du site actif

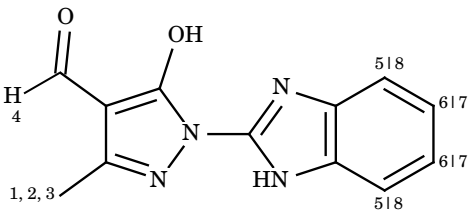


(c) Formule développée plane du CVR161

FIGURE 7.15 – Modèle de liaison du CVR161 au site actif de l'intégrase du VIH-1.



(a) *Epitope mapping* WaterLOGSY du CVR161 – en noir –, comparé avec la structure de *docking* (figure 7.15a page ci-contre) – en gris –



(b) Formule développée plane du CVR161

FIGURE 7.16

résultats sont similaires en considérant les deux systèmes présentés. Les résultats sont présentés sur la figure 7.18 page suivante. La formule du composé CVR224 reste pour l'instant confidentielle. Ainsi, afin de pouvoir présenter la répartition des points de contact sur ce ligand sans révéler la formule chimique, les atomes ont été remplacés par des sphères. Mis à part trois groupes, indiqués en vert sur la figure, l'ensemble des atomes semble impliqué dans l'interaction avec l'intégrase. La répartition des intensités STD relative, sur la structure du CVR224 permet de déterminer l'orientation du ligand sur l'intégrase.

L'arbre présenté sur la figure 7.7 page 160 classe le CVR224 comme un ligand du site W131 - Dimère. Afin d'affiner la prédiction nous pouvons inclure dans le modèle les données d'*epitope mapping* obtenus pour ce ligand en utilisant le protocole présenté dans la partie 5.4 page 106.

Un *blind docking* du CVR224 a été réalisé sur l'ensemble du dimère CCD_2 de l'intégrase avec le logiciel AutoDock 4.0, selon le protocole décrit dans le paragraphe 7.1.2 page 158. Afin d'échantillonner au maximum l'espace conformationnel, cent poses de *docking* ont été générées.

L'arbre obtenu permettant de regrouper les poses de *docking* selon leur similarité d'*epitope mapping* est représenté sur la figure 7.19a page 173. La branche contenant le vecteur représentant les données expérimentales comporte 13 éléments, soit 12 poses de *docking* différentes. Afin de réduire le nombre de poses sélectionnées, les 12 vecteurs correspondant ont été utilisés

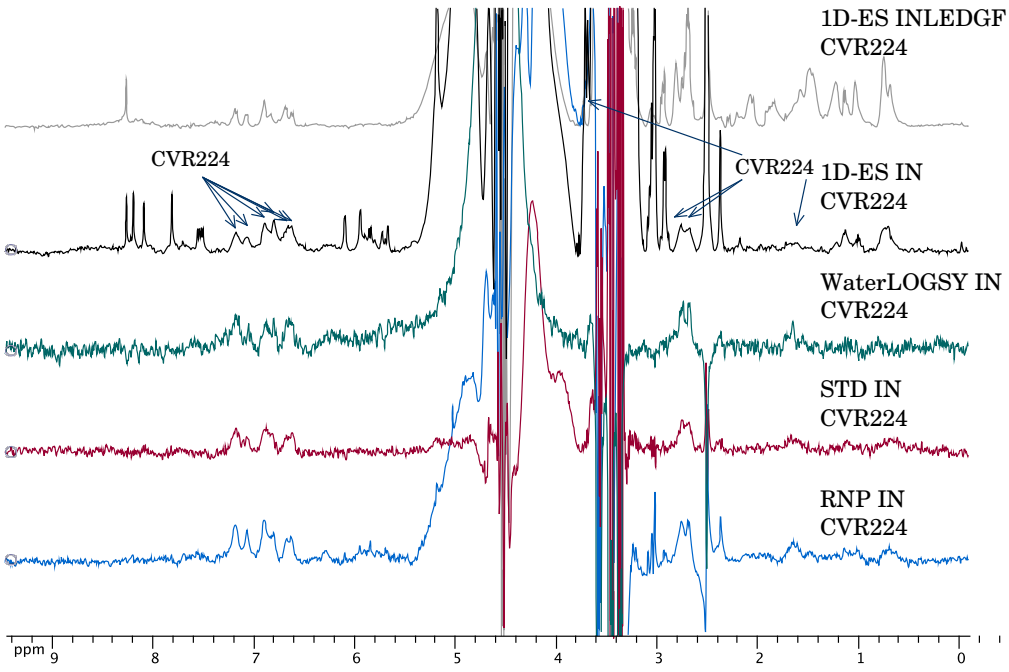


FIGURE 7.17 – Spectres RMN du CVR224 en interaction avec l'intégrase du VIH-1. Abbréviations : 1D-ES : spectre à une dimension utilisant la méthode *excitation-sculpting* pour éliminer l'eau ; INLEDGF : complexe intégrase·LEDGF ; IN : Intégrase du VIH1.

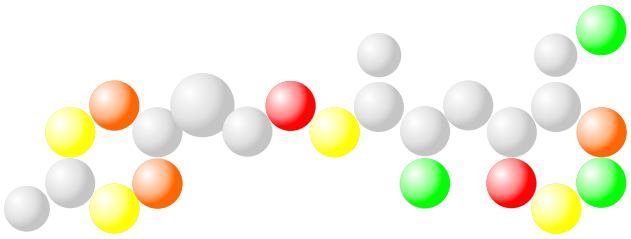
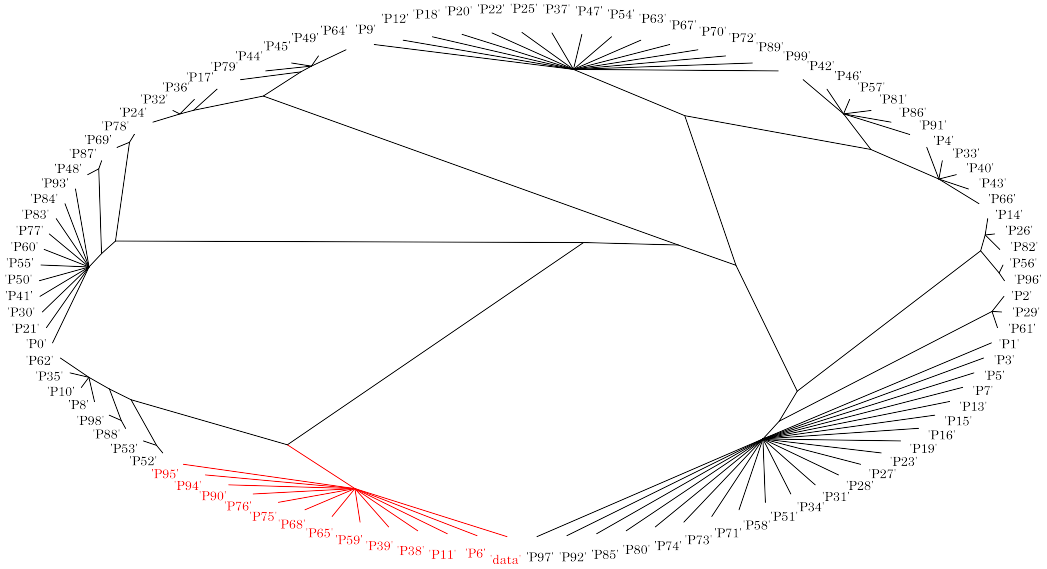
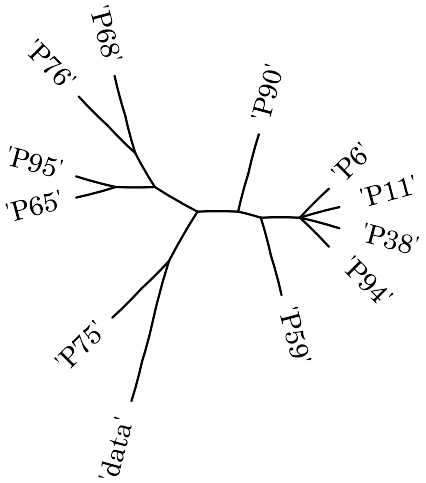


FIGURE 7.18 – *Epitope mapping* STD du CVR224. Le code couleur indique le rapport de l'intensité du signal STD par rapport au signal 1D classique. Rouge : $R > 0,9$; orange : $0,7 < R \leq 0,9$; jaune : $0,6 < R \leq 0,7$; vert : $R \leq 0,6$.



(a) Regroupement de 100 poses de *docking*



(b) Regroupement des 12 poses de *docking* identifiées ci-dessus

FIGURE 7.19 – Regroupement hiérarchique par carte de Kohonen, de dimension 5×4, des poses de *docking* de la molécule CVR224 sur l'intégrase du VIH-1 et comparaison aux données expérimentales notées "data". La branche contenant ces données est mise en évidence par la couleur rouge.

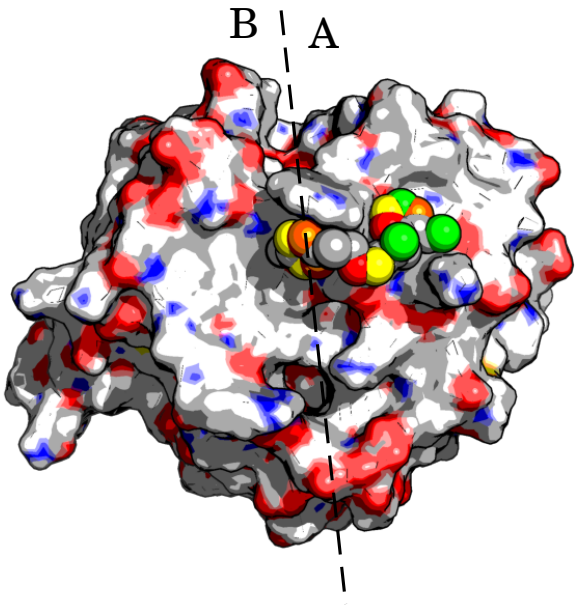


FIGURE 7.20 – Structure de *docking* du CVR224 sélectionnée selon les données expérimentales d'*epitope mapping*. L'*epitope mapping* présenté en figure 7.18 page 172 est reporté sur la structure tridimensionnelle du ligand lié au récepteur.

pour entraîner de nouveaux une carte de Kohonen de dimension 5×4. Le résultat du regroupement est présenté en figure 7.19b page précédente.

La pose 75, la plus proche des données d'*epitope mapping* expérimentales, est représentée sur la figure 7.20. Le composé CVR224 est ici docké sous le tryptophane 131 proche du site d'interaction avec le LEDGF. Ce site de liaison est intéressant pour plusieurs raisons :

- le tryptophane 131 est impliqué dans la liaison au LEDGF en formant de nombreux contacts hydrophobes avec l'IBD et une liaison hydrogène avec l'arginine 405 de l'IBD du LEDGF [23]
- la mutation W131A réduit les capacités de réplication du VIH-1 *in cellulo* [17]
- la mutation W131A n'affecte pas l'activité enzymatique de transfert de brin en l'absence de LEDGF [17]
- l'activité enzymatique n'est pas augmenté pour le mutant W131A lors de l'ajout de LEDGF par rapport à l'enzyme non mutée [17]

Ainsi un phénomène allostérique pourrait entraîner la dissociation du complexe entre l'intégrase et le LEDGF ou au moins réduire l'affinité entre les

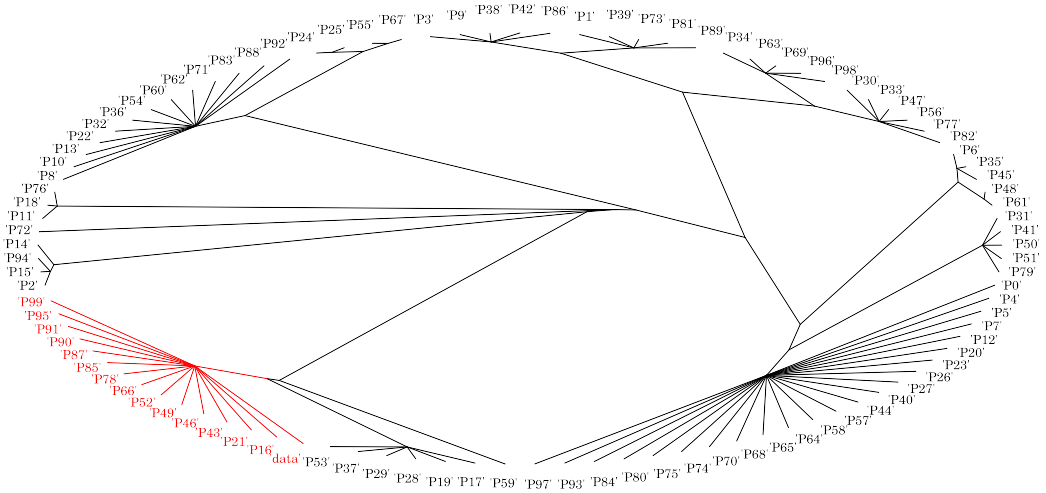
deux protéines.

Cependant, malgré l'intérêt que représente ce site, plusieurs constatations sont à prendre en considération :

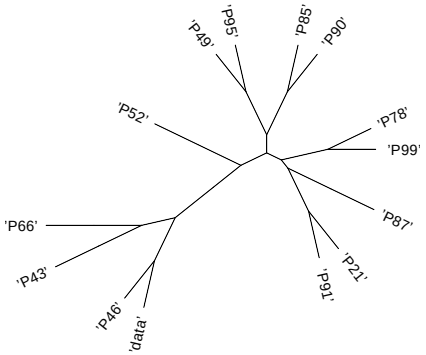
- la corrélation entre l'*epitope mapping* et la pose de *docking* n'est pas bonne comparée aux corrélations observées pour le site catalytique par exemple ; ceci est visible sur l'arbre de la figure 7.19b page 173 : les données expérimentales sont isolées du reste de l'arbre à l'extrémité d'une branche de taille importante ($d = 0,93$ pour une distance totale normalisée à 1)
- la zone prédite pour ce composé d'après l'analyse statistique des contacts n'est pas la zone W131 mais la zone "W131 - Dimère" (voir figure 7.7 page 160), ainsi la pose retenue n'est pas fréquemment obtenue par l'algorithme de *docking*
- la pose de plus basse énergie pour le *docking* considérée ne se trouve pas au niveau du site retenu mais au niveau de la zone "Dimère".

Afin de comprendre ces résultats, il est nécessaire de prendre en considération la mutation de solubilité F185K [82] utilisée pour l'obtention des cristaux d'intégrase pour les expériences de diffraction des rayons X. Cette mutation de solubilité, absente des protéines utilisées pour les expériences RMN, est localisée au niveau du site "Dimère" de l'intégrase et peut donc modifier les résultats de *docking* obtenus pour ce site. Un modèle d'intégrase, généré avec le logiciel Modeller [43], ne présentant pas la mutation F185K a été utilisé afin de réaliser les mêmes expériences de *blind docking* que celles décrites ci-dessus. La structure obtenue sera noté intégrase-WT (*Wild-Type*) dans la suite du manuscrit. Le regroupement des cent poses de *docking* selon leur *epitope mapping* est donné en figure 7.21a page suivante et le regroupement itératif des 13 poses identifiées est représenté en figure 7.21b page suivante. L'analyse du sous arbre montre que la pose 46 est proche des données expérimentales. En comparaison avec les résultats obtenus sur l'intégrase portant la mutation F185K, les données expérimentales ne sont pas exclues dans un neurone unique – représenté par une branche sans ramification – mais sont regroupées avec les données extraites de la pose 46. Ceci montre une plus grande corrélation des données avec la pose 46 générée sur l'intégrase-WT que la pose 75 générée sur l'intégrase portant la mutation F185K. La pose de *docking* sélectionnée est réprésenté sur la figure 7.22 page 177. Un ensemble de données converge vers ce site "Dimère" pour la fixation du CVR224 :

- ce site est énergétiquement favorable à la fixation d'un ligand (voir figure 7.4 page 156)
- la pose du CVR224 sélectionnée correspond à la pose de plus basse énergie



(a) Regroupement de 100 poses de *docking*



(b) Regroupement des 13 poses de *docking* identifiées ci-dessus

FIGURE 7.21 – Regroupement hiérarchique par carte de Kohonen, de dimension 5×4, des poses de *docking* de la molécule CVR224 sur l'intégrase-WT et comparaison aux données expérimentales notées "data". La branche contenant ces données est mise en évidence par la couleur rouge.

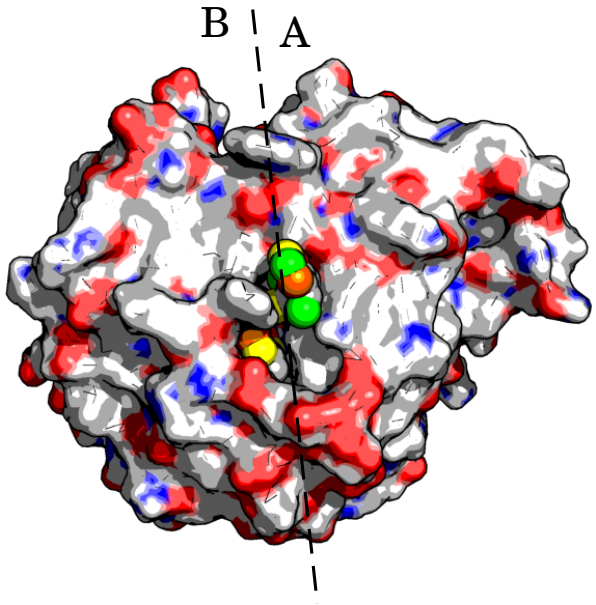


FIGURE 7.22 – Structure de *docking* du CVR224 sur l'intégrase-WT sélectionnée selon les données expérimentales d'*epitope mapping*. L'*epitope mapping* présenté en figure 7.18 page 172 est reporté sur la structure tridimensionnelle du ligand lié au récepteur.

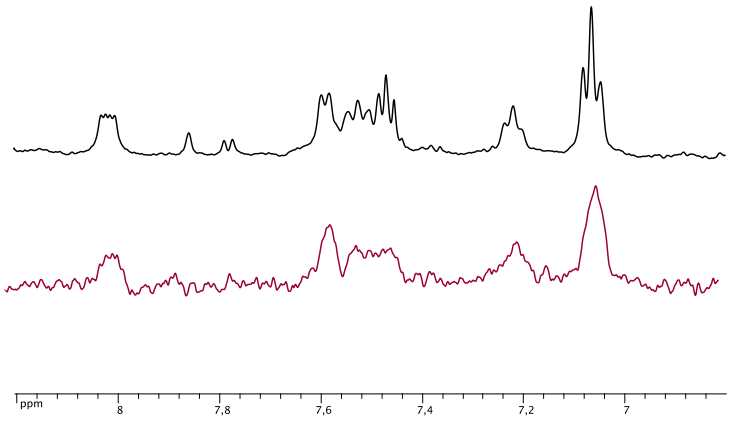
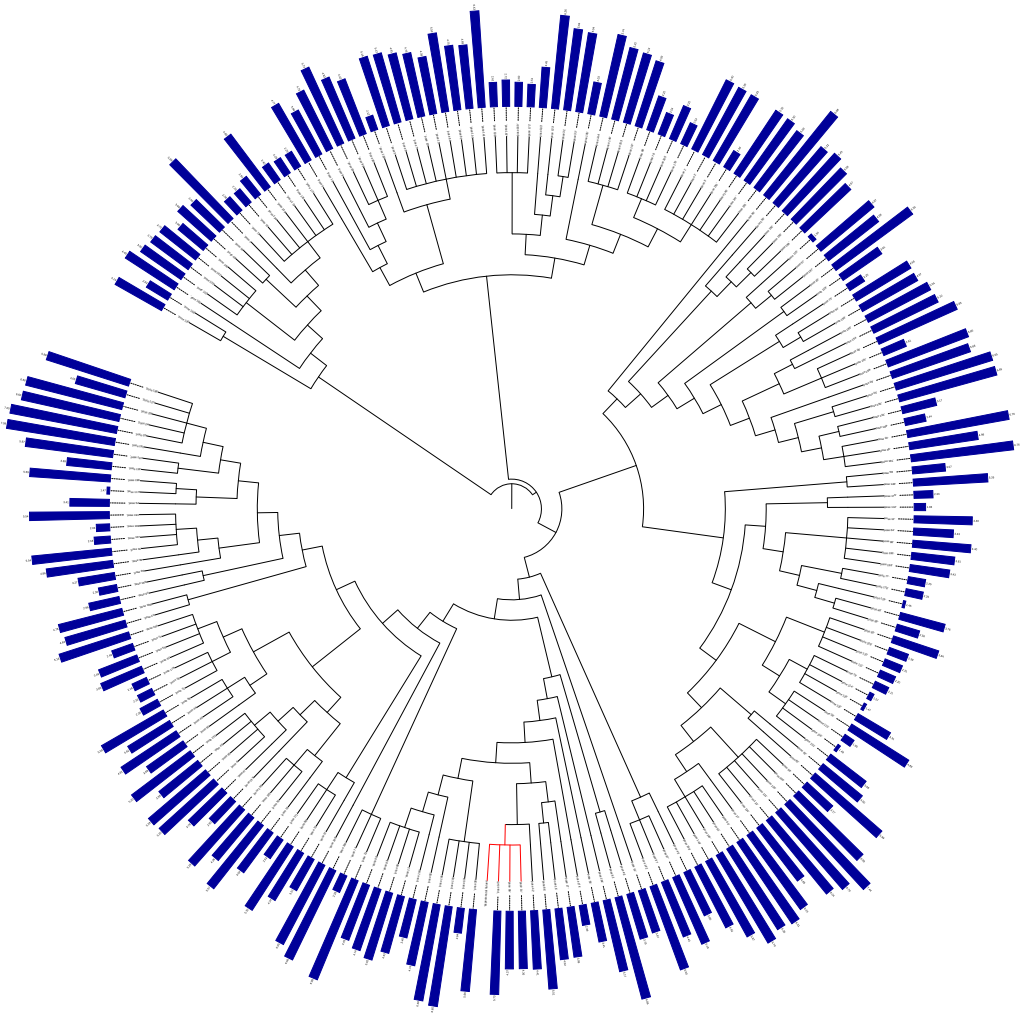


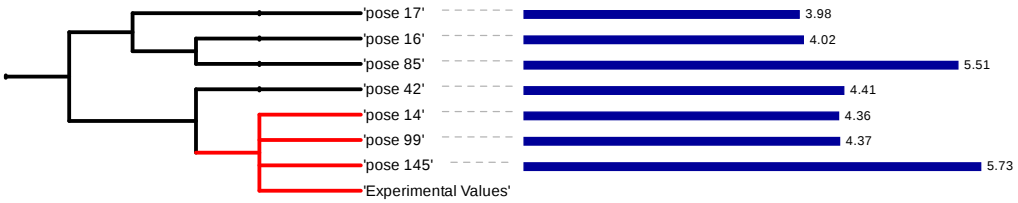
FIGURE 7.23 – Région des déplacements chimiques des protons aromatiques des spectres 1D classique (en noir) et STD (en rouge) du composé CVR16006 en interaction avec le complexe intégrase·LEDGF.

– les données d'*epitope mapping* corrèle bien avec la pose de *docking*
La phénylalanine 185 semble stabiliser le CVR224 en effectuant des interactions hydrophobes avec ce composé. La position du site de liaison au niveau de l'interface entre les deux monomères est cohérente avec l'hypothèse d'un mode d'action allostérique du CVR224 [59]. La lysine 186, proche du site de fixation identifié, est importante à la multimérisation de l'intégrase [13]. Des résultats récents ont montré que la mutation de la lysine 186 en glutamine (K186Q) diminue l'affinité de l'intégrase pour le LEDGF [182]. Cette diminution d'affinité est due à la perte de l'état oligomérique de l'intégrase suite à la mutation K186Q. Le CVR224 pourrait agir sur la déstabilisation des multimères d'intégrase ce phénomène se répercutant sur l'interaction avec le LEDGF.

Le composé CVR16006, classé dans le même groupe que le composé CVR224 (voir figure 7.7 page 160) s'est révélé positif d'après les expériences d'interactions effectuées en RMN (voir figure 7.23). La comparaison entre les poses de *docking* et les données expérimentales d'*epitope mapping* a été réalisée comme pour le CVR224. Pour ce composé, 200 poses de *docking* ont été générées et classées en utilisant une carte de Kohonen de dimension 20×10 (voir figure 7.24 page ci-contre). L'utilisation d'une carte plus grande ici 20×10 contre 5×4, permet d'obtenir directement les poses les plus proches des valeurs expérimentales. Trois poses de *docking* sont proches des données expérimentales (poses 14, 99 et 145). Ces trois poses se retrouvent au niveau du site "Dimère" comme pour le composé CVR224. La pose de plus basse énergie est représentée sur la figure 7.25a page 180. Une bonne corrélation –

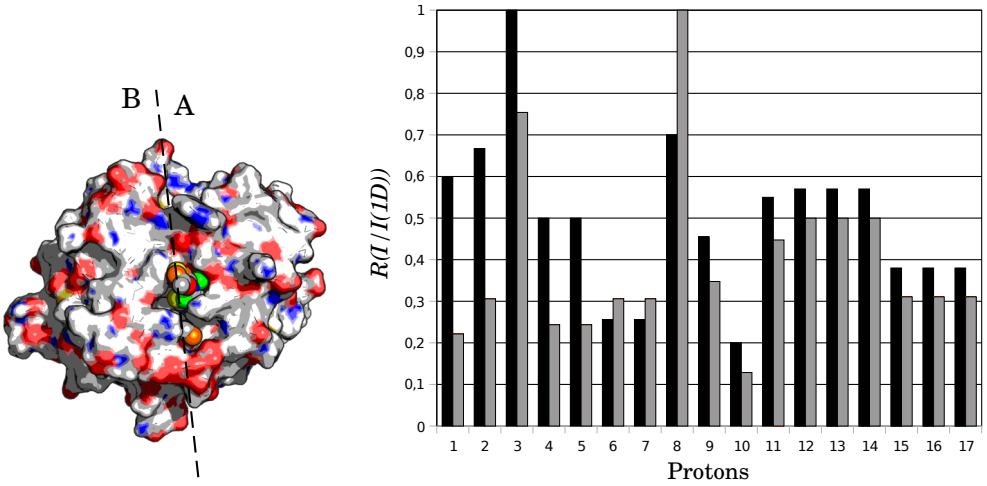


(a) Regroupement de 200 poses de *docking*



(b) Agrandissement de la zone de l'arbre contenant les données expérimentales (*experimental values*)

FIGURE 7.24 – Regroupement hiérarchique par carte de Kohonen, de dimension 20×10, des poses de *docking* de la molécule CVR16006 sur l'intégrase-WT et comparaison aux données expérimentales notées *experimental values*. La branche contenant ces données est mise en évidence par la couleur rouge. L'histogramme bleu donne la valeur calculée du score de *docking* (fonction de score native de Surflex-Dock). Plus cette valeur est grande plus le complexe est stable.



(a) (b)

FIGURE 7.25 – (a) Structure de *docking* du CVR16006 sur l'intégrase-WT sélectionnée selon les données expérimentales d'*epitope mapping*. L'*epitope mapping* est reporté sur la structure tridimensionnelle du ligand lié au récepteur. (b) *Epitope mapping* expérimental (noir) et calculé à partir de la structure de *docking* (gris).

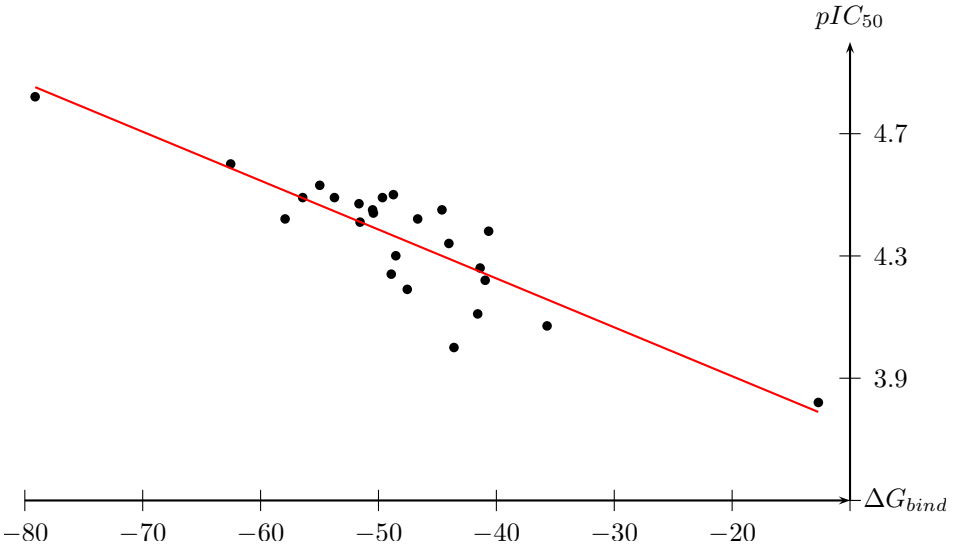


FIGURE 7.26 – Relation entre le $pIC_{50} = -\log(IC_{50})$ de 25 composés CVR et l'énergie libre de liaison ΔG_{bind} calculée par la méthode MM–GBSA (voir paragraphe 3.1.6 page 53) pour les poses de *docking* correspondantes sur le site "Dimère". La droite de régression linéaire, d'équation $y = -1,6 \cdot 10^{-2}x + 3,6$, est donnée pour l'ensemble des 25 points. Le coefficient R^2 est de 0,81.

de 0,68 – est observée entre l'*epitope mapping* expérimental et celui calculé à partir de la structure de *docking* (voir histogramme sur la figure 7.25b page ci-contre).

Différents composés ont ensuite été développés par la société CellVir à partir de la structure du CVR224 et du CVR16006. L'IC_{50} a été déterminée pour l'ensemble de ces composés par des tests de dissociation entre l'IBD du LEDGF et le CCD de l'intégrase suivis par FRET [2]. Les IC_{50} des composés présentant une activité se répartissent sur une gamme allant de 15 à 150 μM. Vingt-cinq composés positifs pour ce test ont été dockés sur le site "Dimère" identifié pour les composés CVR224 et CVR16006. L'énergie d'interaction ΔG_{bind} a été évaluée à partir des structures de *docking* par la méthode MM–GBSA détaillée dans le paragraphe 3.1.6 page 53. La relation entre l'énergie d'interaction calculée et les données expérimentales $pIC_{50} = -\log(IC_{50})$ est donnée sur le graphique de la figure 7.26. Une bonne corrélation ($R^2 = 0,81$) est présente entre les données expérimentales et théoriques calculées par la méthode MM–GBSA. Ce résultat corrobore ceux précédemment obtenus quand à la qualité de la prédiction du site de fixation de cette série de ligands.

2. *Fluorescence Resonance Energy Transfer*

Quatrième partie

Conclusion et perspectives

Chapitre 8

AuPosSOM : conclusion et perspectives

Beaucoup de stratégies de criblage virtuel reposent sur le tri par le score des conformations générées par un logiciel de *docking*. Cependant l'évaluation des énergies d'interaction reste difficile à prédire et les fonctions de score donnent souvent des valeurs "lissées", afin de s'adapter au plus grand nombre de cibles protéiques.

L'utilisation des informations de contacts ligand·récepteur, afin de trier les molécules, permet de s'affranchir du problème de la fonction de score. L'approche développée au sein du logiciel AuPosSOM repose sur l'analyse des contacts entre ligand et récepteur et l'utilisation de ces informations afin de trier de manière automatique et non supervisée les molécules. L'étude de différentes cibles (protéase, transcriptase inverse, intégrase du VIH-1 et thrombine humaine) montre que le tri selon les contacts donne des résultats plus sensibles et plus spécifiques que le tri selon le score.

Cette nouvelle approche utilise l'ensemble des poses générées par le logiciel de *docking*. L'analyse statistique de la répartition des poses apporte la composante entropique à cette méthode. En effet, la fréquence selon laquelle une conformation est retrouvée, est liée au terme entropique de l'énergie de liaison de cette molécule [19]. Une pose qui se retrouve fréquemment est liée à une entropie favorable dans le processus de liaison au récepteur. Cette fréquence se retrouve dans la valeur des éléments du vecteur de la molécule et leur distribution.

La méthode de regroupement des molécules par homologie de contacts utilise les cartes de Kohonen. Le processus itératif de l'algorithme correspondant permet d'éliminer les poses qui ne sont retrouvées que peu fréquemment. La comparaison avec un autre algorithme de *clustering*, de type *k-means*, montre de moins bon résultats pour ce dernier, surtout lorsque les poses sont

dispersées.

Des approches par contact ont aussi été développées par d'autres groupes. L'approche majeure est l'approche SIFt [1] [37]. Cette approche utilise une autre méthode de regroupement des molécules basée sur le coefficient de Tanimoto. Du fait de la méthode de comparaison des vecteurs, l'analyse doit se faire sur une seule pose par molécule, ce qui soulève le problème de la sélection de celle ci. De plus, une molécule témoin, dont la conformation liée au sein du site de fixation est connue, est nécessaire afin de générer une empreinte de référence.

La méthode de regroupement hiérarchique par carte de Kohonen permet aussi la comparaison du mode de fixation de petites molécules organiques par rapport à des peptides ligands. Cette approche a été utilisée avec succès dans la découverte d'inhibiteurs de la β-TrCP, suite du travail de thèse de Julien Pons [144].

Le logiciel AuPosSOM a été valorisé par son dépôt à l'Agence pour la Protection des Programmes (APP) (voir figure 8.1 page suivante).

Le logiciel AuPosSOM est téléchargeable sur le site internet :
http ://www.aupossom.com,
et peut être directement utilisé en ligne sans installation préalable.

L'amélioration de la qualité du regroupement par homologie de contact est en cours de réalisation. Celle ci passe par la prise en considération des liaisons électrostatiques et l'affinement des vecteurs descriptifs des interactions.

1. *Structural Interaction Fingerprints*

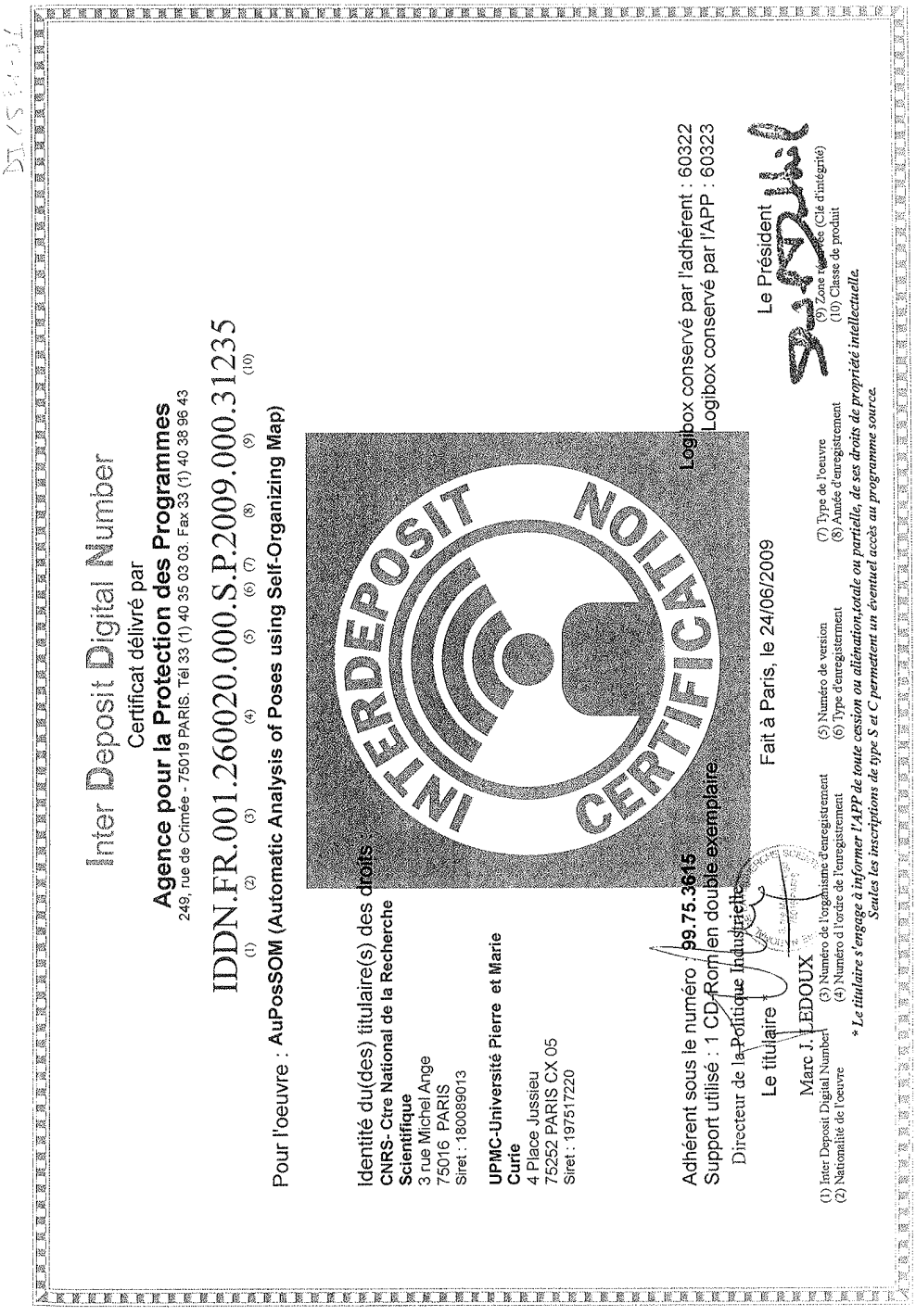


FIGURE 8.1 – Dépôt APP du logiciel AuPosSOM

Chapitre 9

Vers de nouveaux inhibiteurs de l'intégrase du VIH-1

La combinaison de données expérimentales RMN avec des outils de modélisation moléculaire a permis de proposer un site de fixation pour de nouvelles molécules ciblant l'intégrase du VIH-1. Ces molécules possèdent des propriétés inhibitrices de l'interaction entre intégrase et LEDGF. Le site de fixation identifié se trouve à l'interface entre les deux domaines catalytiques de l'intégrase (CCD), à 20 Å de l'interface IBD·CCD (voir figure 9.1 page suivante).

Ainsi l'action de l'inhibiteur sur le LEDGF ne peut se faire par compétition entre l'inhibiteur et l'IBD du LEDGF pour le site sur le dimère CCD_2. Afin de comprendre le mécanisme potentiel de cette inhibition il est nécessaire de considérer l'organisation tétramérique de l'intégrase, celle-ci impliquant ses domaines N-terminaux (NTDs).

La structure de *docking* présentée en figure 9.1 page suivante a été alignée avec la structure de l'intégrase du MVV à deux domaines (NTD-CCD) complexée à l'IBD du LEDGF (code pdb : 3hph, voir figure 9.2a page suivante). L'inhibiteur se retrouve placé à un point crucial pour l'organisation tétramérique de l'intégrase (voir figure 9.2b page suivante). La poche de fixation est en partie confondue avec le site d'interaction NTD·CCD responsable de la tétramérisation de l'intégrase. De plus, cette interaction NTD·CCD est nécessaire à la stabilisation du complexe intégrase·LEDGF et explique la haute affinité du LEDGF pour l'intégrase [108].

De plus le site identifié coïncide avec quatre acides aminés (W132, M178, F181 et F185), impliqués dans une interaction π nécessaire à l'intégration de l'ADN viral et à l'infection par le VIH-1 [3] (voir figure 9.3 page 191).

Le mécanisme d'action envisagé pour ces inhibiteurs est un mode d'action allostérique. L'IBD du LEDGF se lie au dimère CCD_2 de l'intégrase.

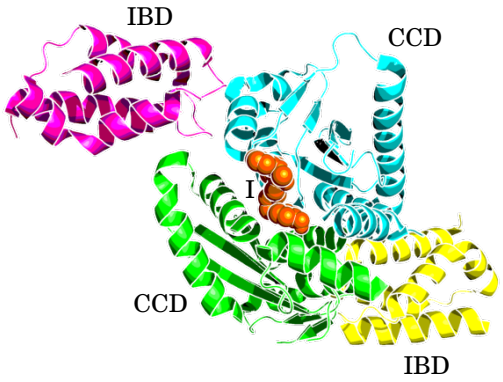
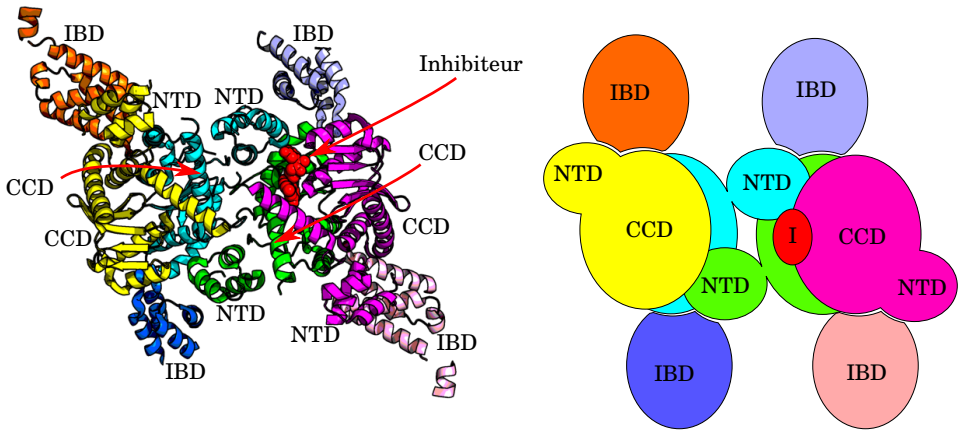


FIGURE 9.1 – Site de fixation présumé des inhibiteurs identifiés comme actif sur la dissociation du complexe IN·LEDGF. Seules les domaines CCD de l'intégrase et IBD du LEDGF sont présents pour la structure du récepteur (code pdb : 2b4j).



(a) (b)

FIGURE 9.2 – (a) : Alignement de la structure de *docking* d'un inhibiteur identifié avec la structure du complexe NTD-CCD·IBD (code pdb : 3hph) de l'intégrase du MVV. (b) : Modèle simplifié de la structure (a).

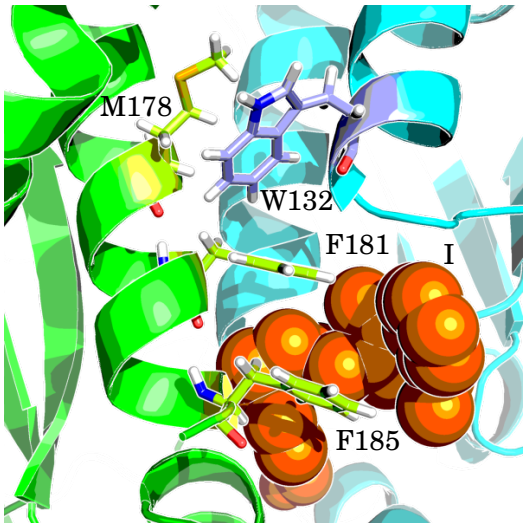


FIGURE 9.3 – Interactions π impliquant les résidus W132, M178, F181 et F185 de l'intégrase du VIH-1. Le monomère A est en vert et le monomère B est en cyan. Un des inhibiteurs identifiés (I) est en orange.

Cette fixation entraînerait l'apparition de l'interface NTD·CCD interdimère [66], ceci expliquant la stimulation de la tétramérisation de l'intégrase par le LEDGF. Ce contact interdimère NTD·CCD stabilise dans un même temps le complexe entre l'intégrase et le LEDGF du fait de l'établissement d'un contact NTD·IBD. Ainsi la fixation d'une molécule inhibitrice au point clé du contact interdimère NTD·CCD empêcherait la formation du tétramère actif et la stabilisation concomitante du complexe entre l'intégrase et le LEDGF.

Cinquième partie

Annexes

Chapitre 10

Codes sources

10.1 Génération des vecteurs

Ensembles de fonctions permettant la génération des vecteurs.

scripts/vector_builder.py

```
1  #! /usr/bin/env python
2  # -*- coding: utf-8 -*-
3  __author__ = " Guillaume Bouvier <guillaume.bouvier@univ-paris5.
       fr> \n Gildas Bertho"
4  __date__ = "01 October 2008"
5  __version__ = "1.0"
6  import os
7  import shutil
8  import glob
9  import h_bond_calc
10 import hydrophobic_contacts
11 import readconf
12 import re
13 from Bio.PDB import *
14 import progressbar
15 import user_interface
16 import autodockScore
17
18 def interleaving_vector(*vectors):
19   """
20   return an interleaved vector of vectors.
21   For exemple : interleaving_vector([1,2,3], [4,5,6], [7,8,9]) =
         [1, 4, 7, 2, 5, 8, 3, 6, 9]
22   """
23   interleaved_vector = []
24   for i in range(len(vectors[0])):
25     for vector in vectors:
26       interleaved_vector.append(vector[i])
```

```
27    return interleaved_vector
28
29  def mean_vector(*vectors):
30    """
31    return a mean vector of the vectors.
32    For exemple : mean_vector([1,2,3], [4,5,6], [7,8,9]) = [4.0,
          5.0, 6.0]
33    """
34    if type(vectors[0]) != tuple:
35      mean_vector = [sum([float(vector[i]) for vector in vectors])
            /len(vectors) for i in range(len(vectors[0]))]
36      return mean_vector
37    else:
38      vectors = vectors[0]
39      mean_vector = [sum([float(vector[i]) for vector in vectors])
            /len(vectors) for i in range(len(vectors[0]))]
40      return mean_vector
41
42  def ligand_vector(ligand_name, h_bond = True, hydrophobic = True
      , minimum = False):
43    """
44    output : interleaved vector : [H,h,H,h,...], where H is an
          Hydrogen bound contact and h an hydrophobic contact
45    """
46    os.chdir(ligand_name)
47    if minimum == False:
48      list_pdb = glob.glob('*_[0-9].pdb')
49      list_pdb.extend(glob.glob('*_[0-9][0-9].pdb'))
50      list_pdb.extend(glob.glob('*_[0-9][0-9][0-9].pdb'))
51      if '%s.pdb' % ligand_name in list_pdb:
52        list_pdb.remove('%s.pdb' % ligand_name)
53    elif minimum == True:
54      structList = autodockScore.readScoreFile(ligand_name, '../
            AuPosSOM_analysis/AuPosSOM.score')
55      list_pdb = []
56      for struct in structList:
57        list_pdb.extend(glob.glob('*_%s.pdb' % struct))
58      if '%s.pdb' % ligand_name in list_pdb:
59        list_pdb.remove('%s.pdb' % ligand_name)
60    h_bonds = ()
61    hydrophobics = ()
62    testvar = True
63    for pdb in list_pdb:
64      parser = PDBParser()
65      try:
66        structure = parser.get_structure('complex', pdb)
67      except:
68        testvar = False
69        print "\n\n\n\nError in structure %s\n" % pdb
```

```
70          vector = 'ERROR'
71          break
72        if h_bond:
73          h_bonds = h_bonds + ( h_bond_calc.make_Hbond_vector(
                h_bond_calc.find_Hbond(structure), structure), )
74        if hydrophobic:
75          hydrophobics = hydrophobics + ( hydrophobic_contacts.
                make_hydrophobic_vector(hydrophobic_contacts.
                find_hydrophobic_contacts(structure), structure), )
76      if h_bond and testvar:
77        mean_h_bonds = mean_vector(h_bonds)
78      if hydrophobic and testvar:
79        mean_hydrophobics = mean_vector(hydrophobics)
80      if h_bond and hydrophobic and testvar:
81        vector = interleaving_vector(mean_h_bonds, mean_hydrophobics
              )
82      elif hydrophobic and testvar and not h_bond:
83        vector = mean_hydrophobics
84      elif h_bond and testvar and not hydrophobic:
85        vector = mean_h_bonds
86      os.chdir('../.')
87      return vector
88
89  def read_ligand_input_file():
90      """
91      return a vector containing the name of ligands
92      """
93      # read configuration file AuPosSOM.conf to obtain the name of
            the file containing ligand names
94      confl = readconf.openconff('AuPosSOM.conf')
95      confl = readconf.extract_conffile(confl, 'Docking_parameters')
96      ligand_input_file_name = readconf.readvalue(confl, '
            ligands_list')
97      ligand_input_file = open('%s' % ligand_input_file_name, 'r')
98      ligands_list = ligand_input_file.readlines()
99      ligand_input_file.close()
100     ligands_list = [ligand.strip() for ligand in ligands_list if
            ligand.strip() != '' and re.findall(r'#', ligand.strip())
            == [] and re.findall(r'<\w*>', ligand.strip()) == []]
101     return ligands_list
102
103 def ALL_vectors(ligands_list, h_bond = True, hydrophobic = True,
        minimum = False):
104     ## Progress bar
105     ligand = 'ligand'
106     widgets = ['Building␣Vectors', progressbar.Percentage(),
            progressbar.Bar(marker='=',left='[',right=']'), progressbar
            .ETA()]
107     pbar = progressbar.ProgressBar(widgets=widgets, maxval=len(
```

```
      ligands_list))
108   pbar.start()
109   barre = 0
110   ##
111   ALL_vectors_list = []
112   for ligand in ligands_list:
113     #widgets = ['', progressbar.Percentage(), progressbar.Bar(
          marker='=',left='[',right=']'), progressbar.ETA()]
114     #pbar = progressbar.ProgressBar(widgets=widgets, maxval=len(
          ligands_list))
115     #ALL_vectors_list.append(ligand_vector(ligand, h_bond,
          hydrophobic, minimum))
116     try:
117       ligvect = ligand_vector(ligand, h_bond, hydrophobic,
            minimum)
118       if ligvect != 'ERROR':
119         ALL_vectors_list.append(ligvect)
120       elif ligvect == 'ERROR':
121 print 'Unable to open pdb file.\nVector generation will stop
    HERE !!! Please re-run AuPosSOM in append mode after having
    corrected the problem.\n
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! VECTOR GENERATION
    ABORTED !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n\n\n\n'
122 break
123     except UnboundLocalError, error:
124       print "\n\n\n\nNo ligand in structure (or ligand name is
            UNK) for: %s\nVector generation will stop HERE !!!
            Please re-run AuPosSOM in append mode after having
            corrected the problem.\n
            !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! VECTOR GENERATION
            ABORTED !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\
            nUnboundLocalError: %s\n\n\n"%(ligand, error)
125       os.chdir('../.')
126       break
127     except OSError, error:
128       print "\n\n\n\nError for ligand %s\nVector generation will
            stop HERE !!! Please re-run AuPosSOM in append mode
            after having corrected the problem.\n
            !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! VECTOR GENERATION
            ABORTED !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\nOSError
            : %s\n\n\n"%(ligand, error)
129       break
130     #except:
131       #print '\n\n\n\nUNKNOWN ERROR (to be corrected) for ligand
            : %s\n!!!!!!!!!!!!!!!!!!!! VECTOR GENERATION ABORTED
            !!!!!!!!!!!!!!!!!!!!!!!!!\n\n\n\n'%ligand
132       ##os.chdir('../.')
133       #break
134     barre = barre + 1
```

```
135         pbar.update(barre)
136       pbar.finish()
137       return ALL_vectors_list
138
139
140     def make_vector_file(h_bond = True, hydrophobic = True, minimum
            = False):
141       ligands_list = read_ligand_input_file()
142       print 'Number of molecules : %s\n' % len(ligands_list)
143       user_interface.makedirectory('AuPosSOM_analysis')
144       ALL_vectors_file, mode = user_interface.open_writeFile('
              AuPosSOM_analysis/ALL_vectors')
145       if mode == 'a':
146         readvectorsfile = open('AuPosSOM_analysis/ALL_vectors', 'r')
147         alreadyDone = [re.split(' ', v)[-1].strip() for v in
                readvectorsfile.readlines() if len(re.split(' ', v)) > 1]
148         for ligand in alreadyDone:
149           ligands_list.remove(ligand)
150         print 'Number of new ligands : %s\n' % len(ligands_list)
151         if len(ligands_list) == 0:
152           os.abort()
153       iter_ligand = iter(ligands_list)
154       ALL_vectors_list = ALL_vectors(ligands_list, h_bond,
              hydrophobic, minimum)
155       if mode != 'a':
156         ALL_vectors_file.write('%s\n' % len(ALL_vectors_list[0]))
157       for vector in ALL_vectors_list:
158         for e in vector:
159           ALL_vectors_file.write('%s ' % e)
160         ALL_vectors_file.write('%s\n' % iter_ligand.next())
161       ALL_vectors_file.close()
162
163     def optimize_vectors(vector_file):
164       shutil.copy(vector_file, 'ALL_vectors.raw')
165       vectorsf = open(vector_file, 'r')
166       vectorsl = vectorsf.readlines()
167       vectorsf.close()
168       os.remove(vector_file)
169       vectorsl = [ re.split(' ', e) for e in vectorsl]
170       numbers = [ [ float(f) for f in e[:-1] ] for e in vectorsl
              ][1:]
171       sum = []
172       for i in range(len(numbers[0])):
173         S = 0
174         for v in numbers:
175           S = S + v[i]
176         sum.append(S)
177       c = 0
178       nonzerosindex = []
```

```
179     for e in sum:
180       if e != 0:
181         nonzerosindex.append(c)
182       c = c +1
183     vectorsf = open(vector_file, 'w')
184     optimizedvectors = []
185     vectorsl = vectorsl[1:]
186     for v in vectorsl:
187       optimizedvector = []
188       for i in nonzerosindex:
189         optimizedvector.append(v[i])
190       optimizedvector.append(v[-1])
191       optimizedvectors.append(optimizedvector)
192     card = len(optimizedvectors[0]) - 1
193     vectorsf.write('%d\n' % card)
194     for v in optimizedvectors:
195       for e in v[:-1]:
196         vectorsf.write('%s ' % e)
197       vectorsf.write('%s' % v[-1])
198
199 def unique(seq):
200         #la fonction permet d'eliminer les elements identiques d
                'une liste
201         checked = []
202         for e in seq:
203                 if e not in checked:
204                         checked.append(e)
205         return checked
206
207 def template_vector(structureFile, r, n):
208     """
209     Make a template vector from a template structure.
          structureFile is a pdb file which is a template. For
          example an Xray structure. r is a radius. Define the area
          in which interactions are calculated.
210     """
211     parser = PDBParser()
212     structure = parser.get_structure('XR', structureFile)
213     atomProtlist = []
214     atomlist = []
215     for model in structure:
216       for chain in model:
217         for residue in chain:
218           if is_aa(residue) == False:
219             ligand = residue
220           else:
221             for atom in residue:
222               atomProtlist.append(atom)
223           for atom in residue:
```

```
224             atomlist.append(atom)
225     sequence = unique([e.get_parent().get_full_id() for e in
            atomProtlist])
226     template_dico = h_bond_calc.FindContact(structure, r)
227     vector = template_dico.values()
228     contacts = []
229     for v in vector:
230       for e in v:
231         seqid = e.get_parent().get_full_id()
232         contacts.append( seqid )
233     contacts = unique(contacts) #amino acids contacted by the
            ligand within a sphere of r angstrom
234     for aa in sequence:
235       if aa in contacts:
236         sequence[sequence.index(aa)] = 1
237       else:
238         sequence[sequence.index(aa)] = 0
239     template_vector = sequence
240     if n == 2:
241       template_vector = interleaving_vector(template_vector,
              template_vector)
242     print 'template vector length : %s ' % len(template_vector)
243     return template_vector
244
245 def multiplyv(v1, v2):
246     c=0
247     m = []
248     for e in v1:
249       m.append(e*v2[c])
250       c = c + 1
251     return m
252
253 def filter_with_template(vector_file, template_vector):
254     """
255     Modify ALL_vectors file to consider only interactions detected
            into template_vector
256     """
257     shutil.copy(vector_file, 'ALL_vectors.raw')
258     vectorsf = open(vector_file, 'r')
259     vectorsl = vectorsf.readlines()
260     vectorsf.close()
261     os.remove(vector_file)
262     vectorsl = [ re.split(' ', e) for e in vectorsl]
263     vectorsl = vectorsl[1:]
264     filtered_vectors = []
265     for v in vectorsl:
266       v1 = v[:-1]
267       v1 = [float(e) for e in v1]
268       v2 = template_vector
```

```
269       m = multiplyv(v1, v2)
270       m.append(v[-1])
271       filtered_vectors.append(m)
272     vectorsf = open(vector_file, 'w')
273     card = len(template_vector)
274     vectorsf.write('%d\n' % card)
275     for v in filtered_vectors:
276       for e in v[:-1]:
277         vectorsf.write('%s ' % e)
278       vectorsf.write('%s' % v[-1])
279     vectorsf.close()
```

Script de calcul des liaisons hydrogènes à partir des coordonnées d'un complexe protéine·ligand au format pdb.

scripts/h_bond_calc.py

```
1  #! /usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Module for h-bonds calculation
5  """
6  __author__ = "Guillaume Bouvier <guillaume.bouvier@univ-paris5.
       fr>\n Gildas Bertho"
7  __date__ = "01 October 2008"
8  __version__ = "1.0"
9
10 from optparse import OptionParser
11 from Bio.PDB import *
12 import readconf
13 import math
14
15 def FindContact(structure, radius):
16   """
17   Find contacts between a ligand and the protein, within a radius
         in angstrom.
18   input :- structure pdb file read by the command :
19     parser = PDBParser()
20     structure = parser.get_structure('complex', filename.pdb)
21    - radius
22   output = {'ligand atom 1' : [atom i, atom j, ...], 'ligand atom
         2' : [atom k, atom l, ...]}, where 'atom x' are atoms of
        the protein.
23   """
24   atomProtlist = []
25   atomlist = []
26   for model in structure:
27     for chain in model:
28       for residue in chain:
29         if is_aa(residue) == False:
```

```
30        if residue.get_resname().strip() == 'Mg':
31          print 'Not yet implemented : Mg element will be ignored'
32        else:
33          ligand = residue
34       else:
35        for atom in residue:
36         atomProtlist.append(atom)
37       for atom in residue:
38        atomlist.append(atom)
39
40   atomLigandlist = []
41   for atomLigand in ligand:
42    atomLigandlist.append(atomLigand)
43   center = NeighborSearch(atomProtlist)
44   contacts = {}
45   for i in range(len(atomLigandlist)):
46    atomLigand = atomLigandlist[i]
47    atomProt = center.search(atomLigandlist[i].get_coord(), radius
         )
48    contact = {atomLigand : atomProt}
49    contacts.update(contact)
50   return contacts
51
52 def find_donors(structure):
53   """
54   If AutoDock has been used, only polar H are presents !!!
55   output : give a list of donors and hydrogen.
56   """
57   HList = []
58   ALLatomlist = []
59   for model in structure:
60    for chain in model:
61     for residue in chain:
62      for atom in residue:
63       ALLatomlist.append(atom)
64       if atom.get_name()[0] == 'H':
65        HList.append(atom)
66
67    removeList, bond = find_Nquat(HList, ALLatomlist)
68    for atoms in removeList:
69     bond.remove(atoms)
70    DH_list = bond
71    DH_list = format_DHList(DH_list)
72    return DH_list
73
74 def format_DHList(DH_list):
75   find_replace_list = []
76   for atoms in DH_list:
77    atomtype = [e.get_name()[0] for e in atoms]
```

```
78      if atomtype.index('H') == 0:
79       find = atoms
80       replace = [atoms[1], atoms[0]]
81       find_replace_list.append([find, replace])
82    for e in find_replace_list:
83     DH_list.remove(e[0])
84     DH_list.append(e[1])
85    return DH_list
86
87  def find_Nquat(HList, ALLatomlist):
88    """
89    Find quaternary nitrogens in an Hydrogen atom list (HList.
90    ALLatomlist is the list of all atoms in the structure
91    ouput : NquatList : list of quaternary Nitrogen, with linked
          atoms.
92     bond : list of all bonded atoms with the H
93    """
94    ALLatomSearch = NeighborSearch(ALLatomlist)
95    bond = []
96    for atom in HList:
97     bond.append(ALLatomSearch.search(atom.get_coord(), 1.5))
98    bondTEMP = bond
99    for e in bond:
100    atomname = [f.get_name()[0] for f in e]
101    if atomname.count('H') == 2:
102     bondTEMP.remove(e)
103     for f in atomname:
104      if f != 'H':
105       try:
106        heteroatom = e[atomname.index(f)]
107        e.remove(heteroatom)
108       except IndexError:
109        print "list index out of range"
110        print "Error in hbond_calc.py : find_Nquat"
111       distances = []
112       distatom = {}
113       for g in e:
114        distance = heteroatom-g
115        distances.append(distance)
116        distatom.update({'%s' % distance : g})
117       bondTEMP.append([heteroatom, distatom['%s' % min(distances)
            ]])
118    bond = bondTEMP
119    removeList = []
120    for atoms in bond:
121     atomstype = [h.get_name()[0] for h in atoms]
122     if atomstype.count('N') >= 1:
123      if len(ALLatomSearch.search(atoms[atomstype.index('N')].
          get_coord(), 1.6)) > 4:
```

```
124         Nquat = atoms
125         removeList.append(Nquat)
126     NquatList = removeList
127     return NquatList, bond
128 
129 
130 def find_bound(atomlist):
131     """
132     Find covalent bounds in an atomlist according to distance
133     """
134     ALLatomSearch = NeighborSearch(atomlist)
135     boundatomsDict = {}
136     for atom in atomlist:
137         boundatoms = ALLatomSearch.search(atom.get_coord(), 1.5)
138         boundatomsDict.update({atom : boundatoms})
139     return boundatomsDict
140 
141 
142 def find_acceptors(structure):
143     NOFSatomlist = []
144     ALLatomlist = []
145     for model in structure:
146         for chain in model:
147             for residue in chain:
148                 for atom in residue:
149                     ALLatomlist.append(atom)
150                     if atom.get_name()[0] == 'O' or atom.get_name()[0] == 'N'
                            or atom.get_name()[0] == 'F' or atom.get_name()[0] == 'S
                            ':
151                         NOFSatomlist.append(atom)
152     ALLatomSearch = NeighborSearch(ALLatomlist)
153     A_list = []
154     for atom in NOFSatomlist:
155         boundatoms = ALLatomSearch.search(atom.get_coord(), 1.8)
156         # find quaternary N and find SO, SO2 and SO3 groups
157         if atom.get_name()[0] == 'O' or atom.get_name()[0] == 'F':
158             A_list.append(atom)
159         else:
160             if not ((('N' in [e.get_name()[0] for e in boundatoms] and
                            len(boundatoms) == 5) == True) or (('S' in [e.get_name()
                            [0] for e in boundatoms] and 'O' in [e.get_name()[0] for e
                             in boundatoms]) == True)):
161                 A_list.append(atom)
162     return A_list
163 
164 def split_A_list(A_list):
165     """
166     Split an acceptors list into two lists : one list for the
            protein and one list for the ligand
```

```
167     """
168     A_list_ligand = []
169     A_list_protein = []
170     for atom in A_list:
171       if is_aa(atom.get_parent()):
172         A_list_protein.append(atom)
173       else:
174         A_list_ligand.append(atom)
175     return A_list_protein, A_list_ligand
176
177   def split_DH_list(DH_list):
178     DH_list_ligand = []
179     DH_list_protein = []
180     for atoms in DH_list:
181       if is_aa(atoms[1].get_parent()):
182         DH_list_protein.append(atoms)
183       else:
184         DH_list_ligand.append(atoms)
185     return DH_list_protein, DH_list_ligand
186
187   def make_dict(DH_list):
188     """
189     Make a dictionary from a DH_list. Format : {'atom ID residue ID
            chain ID' : [D, H]}
190     """
191     DH_dict = {}
192     for e in DH_list:
193       DH_dict.update({'%s_%s_%s' % (e[1], e[1].get_parent(), e[1].
            get_parent().get_parent()) : e})
194     return DH_dict
195
196   def find_Hbond(structure):
197     DH_list = find_donors(structure)
198     A_list = find_acceptors(structure)
199     A_list_protein, A_list_ligand = split_A_list(A_list)
200     DH_list_protein, DH_list_ligand = split_DH_list(DH_list)
201     # read parameters from AuPosSOM.conf file
202     confl = readconf.openconff('../AuPosSOM.conf')
203     confl = readconf.extract_conffile(confl, 'H_bond_calculation')
204     r = float(readconf.readvalue(confl, 'r'))
205     #print 'Ideal distance = %s angstrom' % r
206     D_r_ideal = float(readconf.readvalue(confl, 'D_r_ideal'))
207     D_r_max = float(readconf.readvalue(confl, 'D_r_max'))
208     #print 'Maximum distance deviation = %s angstrom' % D_r_max
209     a_ideal = float(readconf.readvalue(confl, 'a'))
210     #print 'Ideal angle = %s degree' % a_ideal
211     D_a_ideal = float(readconf.readvalue(confl, 'D_a_ideal'))
212     D_a_max = float(readconf.readvalue(confl, 'D_a_max'))
213     #print 'Maximum angle deviation = %s degree' % D_a_max
```

```
214 # DH : prot ; A : ligand
215 DH_dict_protein = make_dict(DH_list_protein)
216 H_list_protein = [e[1] for e in DH_list_protein]
217 #print H_list_protein
218 DH_prot_atomSearch = NeighborSearch(H_list_protein)
219 distance_angle_vector = [ [atom, DH_prot_atomSearch.search(
        atom.get_coord(), r + D_r_max )] for atom in A_list_ligand ]
220 # distance_angle_vector format : [ [atom_ligand, [atom_prot1,
        atom_prot2, ...], [d1, d2]] , ...], where d1 is the distance
         between atom_prot1 and atom_ligand
221 for e in distance_angle_vector:
222  d = []
223  a = []
224  for h in e[1]:
225   # h is an hydrogen of the protein
226   d.append(h - e[0])
227   # find the bounded atom to h
228   key = '%s_%s_%s' % (h, h.get_parent(), h.get_parent().
          get_parent())
229   vectorD = DH_dict_protein[key][0].get_vector()
230   vectorH = DH_dict_protein[key][1].get_vector()
231   vectorA = e[0].get_vector()
232   a.append( (calc_angle(vectorD, vectorH, vectorA))*180/math.pi
          )
233  e.append(d)
234  e.append(a)
235 # DH : ligand ; A : prot
236 DH_dict_ligand = make_dict(DH_list_ligand)
237 H_list_ligand = [e[1] for e in DH_list_ligand]
238 A_prot_atomSearch = NeighborSearch(A_list_protein)
239 distance_angle_vector_ligand = [ [atom, A_prot_atomSearch.
        search( atom.get_coord(), r + D_r_max )] for atom in
        H_list_ligand ]
240
241 for e in distance_angle_vector_ligand:
242  d = []
243  a = []
244  for h in e[1]:
245   # h is an acceptor of the protein
246   # e[0] is an hydrogen of the ligand
247   d.append(h - e[0])
248   # find the bounded atom to h
249   key = '%s_%s_%s' % (e[0], e[0].get_parent(), e[0].get_parent
          ().get_parent())
250   vectorD = DH_dict_ligand[key][0].get_vector()
251   vectorH = DH_dict_ligand[key][1].get_vector()
252   vectorA = h.get_vector()
253   angle = (calc_angle(vectorD, vectorH, vectorA))*180/math.pi
```

```
254     if ( angle > (a_ideal - D_a_max) and angle < (a_ideal +
            D_a_max) ):
255      a.append( angle )
256    e.append(d)
257    e.append(a)
258   distance_angle_vector.extend(distance_angle_vector_ligand)
259   distance_angle_vector = [e for e in distance_angle_vector if e
        [3] != []]
260   #print distance_angle_vector
261   return distance_angle_vector
262
263 def unique(seq):
264         #la fonction permet d'eliminer les elements identiques d
              'une liste
265         checked = []
266         for e in seq:
267                 if e not in checked:
268                         checked.append(e)
269         return checked
270
271 def make_Hbond_vector(distance_angle_vector, structure):
272   sequence = []
273   for model in structure:
274    for chain in model:
275     for residue in chain:
276      if is_aa(residue):
277       sequence.append([residue, chain])
278   residues_Hbonded = []
279   for e in distance_angle_vector:
280    for f in e[1]:
281     residues_Hbonded.append([f.get_parent(), f.get_parent().
          get_parent()])
282   counter_vect = [ [residue, residues_Hbonded.count(residue)] for
        residue in residues_Hbonded ]
283   counter_vect = unique(counter_vect)
284   for e in counter_vect:
285    sequence[sequence.index(e[0])] = e[1]
286   vector = []
287   for e in sequence:
288    if type(e) != int:
289     vector.append(0)
290    else:
291     vector.append(e)
292   return vector
293
294 #parser = OptionParser()
295 #parser.add_option("-f", "--pdbFile", dest="pdb_filename", help
      ="pdb file of the receptor in complex with the ligand",
      metavar="receptor_ligand.pdb")
```

```
296 #(options, args) = parser.parse_args()
297
298
299 #parser = PDBParser()
300 #structure = parser.get_structure('complex', options.
        pdb_filename)
301 #radius = 3
302 ##FindContact(structure, radius)
303
304 #h_bond_list = find_Hbond(structure)
305 #make_Hbond_vector(h_bond_list, structure)
```

Script de calcul des contacts hydrophobes à partir des coordonnées d'un complexe protéine·ligand au format pdb.

scripts/hydrophobic_contacts.py

```
1  #! /usr/bin/env python
2  __author__ = "Guillaume Bouvier <guillaume.bouvier@univ-paris5.
       fr> \n Gildas Bertho"
3  __date__ = "01 October 2008"
4  __version__ = "1.0"
5  from optparse import OptionParser
6  from Bio.PDB import *
7  import h_bond_calc
8  import readconf
9
10 def find_hydrophobic_contacts(structure):
11   """
12   Give hydrophobic contacts between a ligand and a protein in a
         structure.
13   Output : A dictionnary of contacts :
14   {Ligand Atom1 : [ [ [Prot Atom1, Residue, ...], ...], [distance
         Ligand Atom1 - Prot Atom1, ...] ]}
15   """
16   # read parameters from AuPosSOM.conf file
17   confl = readconf.openconff('../AuPosSOM.conf')
18   confl = readconf.extract_conffile(confl, 'hydrophobic_contact')
19   r_hydrophobe = float(readconf.readvalue(confl, 'r_hydrophobe'))
20
21   contacts = h_bond_calc.FindContact(structure, r_hydrophobe)
22   ligatoms = contacts.keys()
23   hydrophobic_contacts = {}
24   for atom in ligatoms:
25     if atom.get_name()[0] == 'C' or atom.get_name()[0] == 'S':
26       contacts[atom] = [e for e in contacts[atom] if (e.get_name()
           [0] == 'C' or e.get_name()[0] == 'S')]
27       contacts[atom] = [[e, e.get_parent()] for e in contacts[atom]
            if is_aa_hydrophobic(e.get_parent())]
28       hydrophobic_contacts.update({atom : contacts[atom]})
```

```
29  lighydrophobic = hydrophobic_contacts.keys()
30  for atom in lighydrophobic:
31   if hydrophobic_contacts[atom] == []:
32    hydrophobic_contacts.pop(atom)
33   else:
34    distances = [ e[0] - atom for e in hydrophobic_contacts[atom]
          ]
35    hydrophobic_contacts[atom] = [hydrophobic_contacts[atom],
         distances]
36  return hydrophobic_contacts
37
38
39 def is_aa_hydrophobic(residue):
40  """
41  return True if the residue is an hydrophobic aa, else return
       False
42  """
43  aa_hydrophobe = ['ALA', 'VAL', 'ILE', 'LEU', 'MET', 'PHE', 'TYR
       ', 'TRP', 'CYS', 'GLY', 'PRO', 'HIS']
44  # Warning : Histidine could be protonated
45  if residue.get_resname() in aa_hydrophobe:
46   return True
47  else:
48   return False
49
50 def unique(seq):
51         #la fonction permet d'eliminer les elements identiques d
               'une liste
52         checked = []
53         for e in seq:
54                 if e not in checked:
55                         checked.append(e)
56         return checked
57
58 def make_hydrophobic_vector(hydrophobic_contacts, structure):
59  sequence = []
60  for model in structure:
61   for chain in model:
62    for residue in chain:
63     if is_aa(residue):
64      sequence.append([residue, chain])
65  contact_residues = []
66  for atom in hydrophobic_contacts.keys():
67   contact_residues.extend( [[e[1], e[1].get_parent()] for e in
        hydrophobic_contacts[atom][0]] )
68  count_contact = []
69  for residue in contact_residues:
70   count_residue = contact_residues.count(residue)
71   count_contact.append([residue, count_residue])
```

```
72  count_contact = unique(count_contact)
73  #print count_contact
74  for e in count_contact:
75   sequence[sequence.index(e[0])] = e[1]
76  #vector = [0 for e in sequence if type(e) != int ]
77  vector = []
78  for e in sequence:
79   if type(e) != int:
80    vector.append(0)
81   else:
82    vector.append(e)
83  return vector
84
85 #parser = OptionParser()
86 #parser.add_option("-f", "--pdbFile", dest="pdb_filename", help
       ="pdb file of the receptor in complex with the ligand",
       metavar="receptor_ligand.pdb")
87 #(options, args) = parser.parse_args()
88
89 #parser = PDBParser()
90 #structure = parser.get_structure('complex', options.
       pdb_filename)
91
92 #hydrophobic_contacts = find_hydrophobic_contacts(structure)
93 #print make_hydrophobic_vector(hydrophobic_contacts, structure)
```

Script de calcul des liaisons hydrogènes et contacts hydrophobes à partir des coordonnées d'un complexe protéine·ligand utilisant des fonctions implémentées au sein du logiciel UCSF-Chimera

scripts/contacts.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # for chimera 1.4
4  from chimera import runCommand
5  rec_file = "rec_charged.mol2"
6  lig_file = "dock_conformers.mol2"
7  #runCommand("open 0 receptor.pdb")
8  runCommand("open 0 %s"%rec_file)
9  runCommand("open 1 %s"%lig_file)
10 runCommand("hbonds intramodel false distSlop 0.65 angleSlop 80
       saveFile hbonds.txt namingStyle serial")
11 runCommand("findclash #0 overlapCutoff 0 saveFile contacts.txt
       namingStyle serial")
12 runCommand("close 1")
13 runCommand("close 0")
14 #for ligNum in range(1, 1001):
15  #runCommand("open 1 ligand.%03d.pdb" % ligNum)
```

```
16  #runCommand("hbonds intramodel false saveFile ligand.%03d.
        hbonds" %
17 #ligNum)
18  #runCommand("close 1")
```

Script permettant la création et la simplification des vecteurs ainsi que le formatage du fichier de vecteurs.

scripts/makeMeanVectors.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  import re
4  import numpy
5  import os
6  #import progressbar
7
8  protein_mol2f = open('rec_charged.mol2', 'r')
9  protein_mol2l = protein_mol2f.readlines()
10 protein_mol2f.close()
11
12 dock_mol2f = open('dock_conformers.mol2', 'r')
13 dock_mol2l = dock_mol2f.readlines()
14 dock_mol2f.close()
15
16
17 def posesCounter(mol_name, dock_names):
18  #nposes = 0
19  nposes = dock_names.count(mol_name)
20  #for e in dock_names:
21   #if e == mol_name:
22    #nposes = nposes + 1
23  return nposes
24
25 def interleaving_vector(*vectors):
26  """
27  return an interleaved vector of vectors.
28  For exemple : interleaving_vector([1,2,3], [4,5,6], [7,8,9]) =
        [1, 4, 7, 2, 5, 8, 3, 6, 9]
29  """
30  interleaved_vector = []
31  n = len(vectors[0])
32  for i in range(len(vectors[0])):
33   #print 'Interleaving vectors: %s/%s'%(i,n)
34   for vector in vectors:
35    interleaved_vector.append(vector[i])
36  return interleaved_vector
37
38 def writeFile(vectors, mol_names):
39  try:
```

```
40      os.mkdir('AuPosSOM_analysis')
41    except OSError:
42      pass
43    outfile = open('AuPosSOM_analysis/ALL_vectors', 'w')
44    outfile.writelines('%s\n'%len(vectors[mol_names[0]]))
45    for mol_name in mol_names:
46      vector = vectors[mol_name]
47      for e in vector:
48        outfile.writelines('%.2f '%e)
49      outfile.writelines('%s\n'%mol_name)
50    outfile.close()
51
52  def optimize_vectors(vectors, mol_names):
53    sumV = [0 for e in range(len(vectors[mol_names[0]]))]
54    lv1 = len(vectors[mol_names[0]])
55    n = len(vectors[mol_names[0]])
56    for i in range(len(vectors[mol_names[0]])):
57      print 'Simplifying vectors: %s/%s '%(i+1,2*n)
58      for mol_name in mol_names:
59        sumV[i] = sumV[i] + vectors[mol_name][i]
60    iv = []
61    for i in range(len(sumV)):
62      print 'Simplifying vectors: %s/%s '%(n+i+1,2*n)
63      if sumV[i] == 0:
64        iv.append(i)
65    for mol_name in mol_names:
66      vectors[mol_name] = numpy.delete(vectors[mol_name], iv)
67    lv2 = len(vectors[mol_names[0]])
68    r = float(lv2)/float(lv1)
69    print 'Vectors simplified (ratio: %s) '%r
70    #i = 0
71    #while i < len(sumV):
72      #print 'Simplifying vectors: %s/%s '%(i,len(sumV))
73      #if sumV[i] == 0:
74        #sumV.pop(i)
75        #for mol_name in mol_names:
76          ##vectors[mol_name].pop(i)
77          #numpy.delete(vectors[mol_name], i)
78      #else:
79        #i = i + 1
80
81
82  # read and extract informations from protein mol2 file
83  for line in protein_mol2l:
84    if re.findall('@<TRIPOS>MOLECULE', line):
85      #print line
86      i = protein_mol2l.index(line)
87  mol_name = protein_mol2l[i+1].strip()
```

```
88  num_atoms = int(protein_mol2l[i+2].strip().split('␣')[0].strip()
        )
89  mol_type = protein_mol2l[i+3].strip()
90  charge_type = protein_mol2l[i+4].strip()
91  print mol_name, num_atoms, mol_type, charge_type
92  del protein_mol2l
93
94  # read and extract the total number of docking (num_docks) and
        the total number of molecules (num_mols) and the names of the
         molecules (mol_names[]), and the name of the molecules with
        docking redundancy (dock_names[])
95  num_docks = 0
96  mol_names = []
97  dock_names = []
98  i = -1
99  for line in dock_mol2l:
100  i = i + 1
101  if re.findall('@<TRIPOS>MOLECULE', line):
102   num_docks = num_docks + 1
103   mol_name = dock_mol2l[i+1].strip()
104   dock_names.append(mol_name)
105   if mol_name not in mol_names:
106    mol_names.append(mol_name)
107 #print num_docks
108 num_mols = len(mol_names)
109 del dock_mol2l
110
111 # create 'zero vectors' for each ligand --- h-bonds (hb) and
        hydrophobics (hp)---
112 hb_vectors = {}
113 #hp_vectors = {}
114 for mol_name in mol_names:
115  hb_vectors.update({mol_name: numpy.zeros(num_atoms)})
116  #hp_vectors.update({mol_name: numpy.zeros(num_atoms)})
117 hp_vectors = hb_vectors
118
119 hbv = []
120 test = False
121 ## Progress bar
122 #widgets = ['h-bonds processing: ', progressbar.Percentage(),
        progressbar.Bar(marker='=',left='[',right=']'), progressbar.
        ETA()]
123 #pbar = progressbar.ProgressBar(widgets=widgets, maxval=len(hbl)
        )
124 #pbar.start()
125 ###
126
127 hbf = open('hbonds.txt', 'r')
128 hbl = hbf.readlines()
```

```
129 hbf.close()
130
131 print 'Reading H-bonds ...'
132 for line in hbl:
133  if re.findall(r'H\-bonds \(donor, acceptor, hydrogen, D\.\.A
        dist, D\-H\.\.A dist\)', line):
134   #print line
135   test = True
136  #print test
137  if test:
138   hbv.append(re.split(r'\s+', line))
139   try:
140    lig_num = int([e for e in re.split(r'\s+', line) if re.
         findall('#1', e)][0].split('.')[1])
141    atom_num = int(re.split(r'\s+', line)[[re.split(r'\s+', line)
         .index(e) for e in re.split(r'\s+', line) if re.findall('
         #0', e)][0] + 1]) # after champagne and martini !
142    da_dist = float(re.split(r'\s+', line)[6]) # distance between
          donor and acceptor
143    lig_name = dock_names[lig_num]
144    #n_poses = posesCounter(lig_name, dock_names)
145    #print lig_name, n_poses, atom_num, da_dist
146    hb_vectors[lig_name][atom_num] = hb_vectors[lig_name][
         atom_num] + 1
147   except IndexError:
148    pass
149  #pbar.update(hbl.index(line))
150 #pbar.finish()
151 del hbl
152
153
154 ## create 'zero vectors' for each ligand --- hydrophobic contacts
       ---
155 #hp_vectors = {}
156 #for mol_name in mol_names:
157  #hp_vectors.update({mol_name: numpy.zeros(num_atoms)})
158
159 hpv = []
160 ## Progress bar
161 #widgets = ['hydrophobic contacts processing: ', progressbar.
     Percentage(), progressbar.Bar(marker='=',left='[',right=']'),
      progressbar.ETA()]
162 #pbar = progressbar.ProgressBar(widgets=widgets, maxval=len(hpl)
     )
163 #pbar.start()
164 ###
165
166 hpf = open('contacts.txt', 'r')
167 hpl = hpf.readlines()
```

```
168 hpf.close()
169
170 print 'Reading contacts ...'
171 for line in hpl:
172  try:
173   #print [e for e in re.split(r'\s+', line) if re.findall('#1',
          e)][0]
174   lig_num = int([e for e in re.split(r'\s+', line) if re.findall
          ('#1', e)][0].split('.')[1])
175   atom_num = int(re.split(r'\s+', line)[[re.split(r'\s+', line).
          index(e) for e in re.split(r'\s+', line) if re.findall('#0'
          , e)][0] + 1]) # after champagne and martini !
176   lig_name = dock_names[lig_num]
177   #n_poses = posesCounter(lig_name, dock_names)
178   #print lig_name, atom_num
179   hp_vectors[lig_name][atom_num] = hp_vectors[lig_name][atom_num
          ] + 1
180  except IndexError:
181   pass
182  #lig_num = int([e for e in re.split(r'\s+', line) if re.findall
       ('#1', e)][0].split('.')[1])
183  #print lig_num
184  #pbar.update(hpl.index(line))
185 #pbar.finish()
186 del hpl
187
188 hbhp_vectors = {}
189 print 'Interleaving vectors ...'
190 for mol_name in mol_names:
191  nposes = posesCounter(mol_name, dock_names)
192  #hb_vectors[mol_name] = [e/nposes for e in hb_vectors[mol_name
       ]]
193  hb_vectors[mol_name] = hb_vectors[mol_name]/nposes
194  #hp_vectors[mol_name] = [e/nposes for e in hp_vectors[mol_name
       ]]
195  hp_vectors[mol_name] = hp_vectors[mol_name]/nposes
196  hbhp_vectors.update({mol_name: interleaving_vector(hb_vectors[
       mol_name], hp_vectors[mol_name])})
197  #hbhp_vectors.update( {mol_name: numpy.concatenate( (hb_vectors
       [mol_name], hp_vectors[mol_name]) )} )
198  #hb_vectors.pop(mol_name) # for memory purpose
199  #hp_vectors.pop(mol_name)
200 #print hbv
201 #print hb_vectors[lig_name][800:1000]
202 #print hp_vectors[lig_name][800:1000]
203 #print hbhp_vectors[lig_name][800:1000]
204 print 'Simplifying vectors ...'
205 optimize_vectors(hbhp_vectors, mol_names)
206 print 'Writing file ...'
```

```
207 writeFile(hbhp_vectors, mol_names)
```

10.2 Carte de Kohonen

Implémentation python de l'algorithme d'apprentissage des cartes de Kohonen.

scripts/SOM.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  import numpy
4  import RandomArray
5  import re
6  import math
7  #import matplotlib.pyplot
8  import random
9  import progressbar
10 import cPickle
11 import ROCSOM
12
13 # Import Psyco if available
14 try:
15   import psyco
16   psyco.full()
17 except ImportError:
18   print 'psyco not available'
19   pass
20
21
22 class SOM:
23
24   def __init__(self, filename="input.dat", confname="AuPosSOM.
       conf",simplify_vectors=False):
25     inputfile = open(filename, 'r')
26     lines = []
27     for line in inputfile:
28       lines.append(line.split())
29     self.inputvectors = [v[0:-1] for v in lines[1:]] # Input
         vectors
30     output = []
31     for v in self.inputvectors:
32       out = []
33       for e in v:
34 out.append(float(e))
35       output.append(out)
36     self.inputvectors = output
```

```
37      self.inputnames = [v[-1] for v in lines[1:]] # Vectors names
38      self.cardinal = int(lines[0][0]) # Number of element for
            each vector
39      inputfile.close()
40      conffile = open(confname, 'r')
41      lines = conffile.readlines()
42      conffile.close()
43      test = False
44      for line in lines:
45        if re.findall('<Docking_parameters>', line):
46  test = True
47        if test:
48  if re.findall('#', line):
49            line = line.split('#')[0]
50  if re.findall(r'ligands_list\s*=', line):
51    self.AuPosSOM_inputFile = line.split('=')[1].strip() #
          finding the name of the AuPosSOM.inp file
52      test = False
53      for line in lines:
54        if re.findall('<TreeSOM>', line):
55          test = True
56        if test:
57          if re.findall('#', line):
58            line = line.split('#')[0]
59          if re.findall(r'X\s*=', line):
60            self.X = int(line.split('=')[1]) # Number of neurons
                in X dimension
61          if re.findall(r'Y\s*=', line):
62            self.Y = int(line.split('=')[1]) # Number of neurons
                in Y dimension
63          if re.findall(r'number_of_phase\s*=', line):
64            self.number_of_phase = int(line.split('=')[1]) #
                Number of training phase
65      i = 1
66      self.alpha_begin = []
67      self.alpha_end = []
68      self.radius_begin = []
69      self.radius_end = []
70      self.iterations = []
71      while i <= self.number_of_phase:
72        test = False
73        for line in lines:
74          if re.findall('<TreeSOM>', line):
75            test = True
76          if test:
77            if re.findall(r'#', line):
78              line = line.split('#')[0]
79            if re.findall(r'alpha_begin_%s\s*='%i, line):
80              self.alpha_begin.append(float(line.split('=')[1]))
```

```
81              if re.findall(r'alpha_end_%s\s*='%i, line):
82                self.alpha_end.append(float(line.split('=')[1]))
83              if re.findall(r'radius_begin_%s\s*='%i, line):
84                self.radius_begin.append(float(line.split('=')[1]))
85              if re.findall(r'radius_end_%s\s*='%i, line):
86                self.radius_end.append(float(line.split('=')[1]))
87              if re.findall(r'iterations_%s\s*='%i, line):
88                self.iterations.append(int(line.split('=')[1]))
89          i=i+1
90        # Vector simplification
91        if simplify_vectors:
92          self.inputvectors = self.simplifyVectors()
93          self.cardinal = len(self.inputvectors[0])
94        # Matrix initialization
95        maxinpvalue = max([max(v) for v in self.inputvectors])
96        mininpvalue = min([min(v) for v in self.inputvectors])
97        self.M = RandomArray.uniform(mininpvalue,maxinpvalue,(self.X
              , self.Y, self.cardinal))
98
99      def loadMap(self, MapFile):
100       MapFileFile = open(MapFile, 'r')
101       self.Map = cPickle.load(MapFileFile)
102       MapFileFile.close()
103       return self.Map
104
105     def makeSubInputvectors(self, n):
106       """
107       Make a self.inputvectors variable with all known ligands and
              a random set of n unknown ligands
108       """
109       rocsom = ROCSOM.ROCSOM(MapFile = 'map_%sx%s.dat' % (self.X,
              self.Y))
110       KL = rocsom.KL
111       UL = rocsom.UL
112       rmL = random.sample(UL, len(UL)-n) # list of randomly chosen
              ligands to remove
113       for U in rmL:
114         self.inputvectors.pop(self.inputnames.index(U))
115         self.inputnames.remove(U)
116
117     def makeNewInputvectors(self, ligands_list):
118       """
119       Make a new input vectors variable with all ligands contains
              in ligands_list
120       """
121       for L in self.inputnames:
122         if L not in ligands_list:
123   self.inputvectors.pop(self.inputnames.index(L))
124   self.inputnames.remove(L)
```

```
125
126   def simplifyVectors(self):
127     """
128       Remove systematic zeros in vectors
129     """
130     sumV = []
131     for Eindex in range(self.cardinal):
132       S = 0
133       for Vindex in range(len(self.inputvectors)):
134 S = S + self.inputvectors[Vindex][Eindex]
135       sumV.append(S)
136     c = 0
137     nonzerosindex = []
138     for e in sumV:
139       if e!=0:
140 nonzerosindex.append(c)
141       c = c + 1
142     simplifiedVectors = []
143     for V in self.inputvectors:
144       simplifiedVector = [V[i] for i in nonzerosindex]
145       simplifiedVectors.append(simplifiedVector)
146     self.simplifiedVectors = simplifiedVectors
147     return self.simplifiedVectors
148
149   def findBMU(self, k, Map):
150     """
151       Find the Best Matching Unit for the input vector number k
152     """
153     V=numpy.ones((self.X,self.Y,self.cardinal))*self.
          inputvectors[k]
154     distmat = numpy.sum( (V - Map) * ( (V-Map)* (numpy.ones([
          self.X,self.Y,self.cardinal])*numpy.ones([1,self.cardinal
          ])) ) , axis=2 )**0.5
155     self.BMUindices = [e[0] for e in numpy.where(distmat==
          distmat.min())]
156     return self.BMUindices
157
158   def radiusFunction(self, t, trainingPhase=0):
159     timeCte = float(self.iterations[trainingPhase])/10
160     self.radius = ( self.radius_begin[trainingPhase] - self.
          radius_end[trainingPhase] ) * math.exp( -t/timeCte ) +
          self.radius_end[trainingPhase]
161     return self.radius
162
163   def learningRate(self, t, trainingPhase=0):
164     timeCte = float(self.iterations[trainingPhase])/10
165     self.learning = ( self.alpha_begin[trainingPhase] - self.
          alpha_end[trainingPhase] ) * math.exp( -t/timeCte ) +
          self.alpha_end[trainingPhase]
```

```
166     return self.learning
167
168   def BMUneighbourhood(self, t, i, j, BMUindices, trainingPhase,
          Map):
169     dist = ( (i - BMUindices[0])**2 + (j - BMUindices[1])**2 )
          ** 0.5
170     self.neighbourhood = math.exp( - dist**2 / ( 2*(self.
          radiusFunction(t, trainingPhase)**2) ) )
171     return self.neighbourhood
172
173   def adjustment(self, k, t, trainingPhase, Map, BMUindices):
174     self.adjustMap = numpy.zeros((self.X, self.Y, self.cardinal)
          )
175     learning = self.learningRate(t, trainingPhase)
176     for i in range(self.X):
177       for j in range(self.Y):
178         W = Map[i,j]
179         neighbourhood = self.BMUneighbourhood(t, i, j,
              BMUindices, trainingPhase, Map)
180         self.adjustMap[i,j] = neighbourhood * learning * (self.
              inputvectors[k] - W)
181     return self.adjustMap
182
183   def learn(self):
184     Map = self.M
185     kv = range(len(self.inputvectors))
186     print 'Learning for %s vectors'%len(self.inputvectors)
187     for trainingPhase in range(self.number_of_phase):
188       ## Progress bar
189       tpn = trainingPhase + 1
190       widgets = ['Training phase %s: ' % tpn, progressbar.
            Percentage(), progressbar.Bar(marker='=',left='[',right
            =']'), progressbar.ETA()]
191       pbar = progressbar.ProgressBar(widgets=widgets, maxval=
            self.iterations[trainingPhase]-1)
192       pbar.start()
193       ###
194       for t in range(self.iterations[trainingPhase]):
195 try:
196   k = random.choice(kv)
197   kv.remove(k)
198 except IndexError:
199   kv = range(len(self.inputvectors))
200   k = random.choice(kv)
201   kv.remove(k)
202 Map = Map + self.adjustment(k, t, trainingPhase, Map, self.
      findBMU(k, Map))
203 pbar.update(t)
204       pbar.finish()
```

```
205         self.Map = Map
206         MapFile = open('map_%sx%s.dat' % (self.X, self.Y), 'w')
207         cPickle.dump(Map, MapFile) # Write Map into file map.dat
208         MapFile.close()
209         return self.Map
210 
211       def distmapPlot(self,k,Map):
212         V=numpy.ones((self.X, self.Y, self.cardinal))*self.
              inputvectors[k]
213         distmat = numpy.sum( (V - Map) * ( (V-Map)* (numpy.ones([
              self.X, self.Y, self.cardinal])*numpy.ones([1, self.cardinal
              ])) ) , axis=2 )**0.5
214         #matplotlib.pyplot.imshow(numpy.ones((self.X, self.Y))-
              distmat/distmat.max(), interpolation='nearest') #
              Normalized Map for color plot
215         #matplotlib.pyplot.show()
216 
217       #def mapPlot(self,Map):
218         #matplotlib.pyplot.imshow(Map/Map.max(), interpolation='
              nearest') # Normalized Map for color plot
219         #matplotlib.pyplot.show()
220 
221       def borderline(self,Map):
222         borderMat = numpy.zeros((self.X*2,self.Y*2))
223         initMat = numpy.repeat(numpy.repeat(Map,2,axis=0),2,axis=1)
224         for i in range(0,self.X*2,2):
225           for j in range(0,self.Y*2,2):
226 if j > 1:
227     borderMat[i,j] = ( numpy.dot( initMat[i,j]-initMat[i,j-1],
          numpy.transpose( initMat[i,j]-initMat[i,j-1] ) ) )**0.5
228 else:
229     borderMat[i,j] = 0
230       for i in range(0,self.X*2,2):
231         for j in range(1,self.Y*2,2):
232 if i > 1:
233     borderMat[i,j] = ( numpy.dot( initMat[i,j]-initMat[i-1,j],
          numpy.transpose( initMat[i,j]-initMat[i-1,j] ) ) )**0.5
234 else:
235     borderMat[i,j] = 0
236       for i in range(1,self.X*2,2):
237         for j in range(0,self.Y*2,2):
238 try:
239     borderMat[i,j] = ( numpy.dot( initMat[i,j]-initMat[i+1,j],
          numpy.transpose( initMat[i,j]-initMat[i+1,j] ) ) )**0.5
240 except IndexError:
241     #borderMat[i,j] = ( numpy.dot( initMat[i,j]-initMat[0,j],
          numpy.transpose( initMat[i,j]-initMat[0,j] ) ) )**0.5
242     borderMat[i,j] = 0
243       for i in range(1,self.X*2,2):
```

```
244         for j in range(1,self.Y*2,2):
245   try:
246     borderMat[i,j] = ( numpy.dot( initMat[i,j]-initMat[i,j+1],
          numpy.transpose( initMat[i,j]-initMat[i,j+1] ) ) )**0.5
247   except IndexError:
248     #borderMat[i,j] = ( numpy.dot( initMat[i,j]-initMat[i,0],
          numpy.transpose( initMat[i,j]-initMat[i,0] ) ) )**0.5
249     borderMat[i,j] = 0
250      #self.borderMat = borderMat[1:self.X*2,1:self.Y*2]/borderMat
           [1:self.X*2,1:self.Y*2].max()
251      self.borderMat = borderMat/borderMat.max() #Normalize
           borderMat values between 0 and 1
252      return self.borderMat
253
254    def clusterDiscovery(self,Map,T):
255      borderMat = self.borderline(Map)
256      Nv = [[i,j] for i in range(self.X) for j in range(self.Y)]
257      random.shuffle(Nv)
258      Cv = []
259      while Nv != []:
260        N = Nv[0] # locate an arbitrary node N
261        C = [] # start a new cluster C
262        A = [] # Adjacent nodes
263        A, C, Nv = self.cluster(borderMat,Nv,N,T,C,A) # call
             procedure cluster for N, C and T
264        while A != []:
265   N = A[0]
266   A.remove(N)
267   A, C, Nv = self.cluster(borderMat,Nv,N,T,C,A) # call procedure
       cluster for N, C and T
268        Cv.append(C)
269      clustersMap = numpy.zeros((self.X,self.Y))
270      CId = 0
271      for v in Cv:
272        CId = CId + 1
273        for e in v:
274   clustersMap[e[0],e[1]] = CId
275      self.clustersMap = clustersMap
276      return self.clustersMap
277
278    def cluster(self,borderMat,Nv,N,T,C,A):
279      C.append(N) # assign N to C
280      #print 'Nv%s'%Nv
281      Nv.remove(N) # mark N as visited
282      c = 0
283      for i in range(N[0]*2,N[0]*2+2):
284        for j in range(N[1]*2,N[1]*2+2):
285   c = c + 1
286   if borderMat[i,j] <= T: # distance < Threshold
```

```
287     if (c == 1 and N[1] >= 1):
288       fc = [N[0],N[1]-1]
289       if (fc in Nv and fc not in A): # unvisited node
290         A.append(fc)
291     if (c == 2 and N[0] >= 1):
292       fc = [N[0]-1,N[1]]
293       if (fc in Nv and fc not in A): # unvisited node
294         A.append(fc)
295     if (c == 3 and N[0] <= self.X - 1):
296       fc = [N[0]+1,N[1]]
297       if (fc in Nv and fc not in A): # unvisited node
298         A.append(fc)
299     if (c == 4 and N[1] <= self.Y - 1):
300       fc = [N[0],N[1]+1]
301       if (fc in Nv and fc not in A): # unvisited node
302         A.append(fc)
303      return A, C, Nv
304
305   def clusterLooseness(self, clustersMap, Map):
306     """
307     return an self.X * self.Y matrix with cluster containing
          cluster looseness values
308     """
309     clusterLoosenessMat = numpy.zeros((self.X, self.Y))
310     for CId in range(int(clustersMap.min()), int(clustersMap.max
          () + 1)):
311       C = [[numpy.where(clustersMap == CId)[0][i], numpy.where(
            clustersMap == CId)[1][i]] for i in range(len(numpy.
            where(clustersMap == CId)[0]))]
312       loosenessMat = numpy.zeros((len(C),len(C)))
313       i = 0
314       for N1 in C:
315   j = 0
316   for N2 in C:
317     loosenessMat[i,j] = (numpy.dot( Map[N1[0],N1[1]]-Map[N2[0],N2
          [1]], numpy.transpose(Map[N1[0],N1[1]]-Map[N2[0],N2[1]]) )
          )**0.5
318     j = j + 1
319   i = i + 1
320       #print loosenessMat.sum()
321       if len(C) > 1:
322   looseness = (loosenessMat.sum()/2) / ( len(C)*(len(C) - 1)/2 )
323       else:
324   looseness = loosenessMat.sum()/2
325       for N in C:
326     clusterLoosenessMat[N[0],N[1]] = looseness
327     self.clusterLoosenessMat = clusterLoosenessMat
328     return self.clusterLoosenessMat
329
```

```
330   def clusterDistance(self, clustersMap, Map):
331     m = int(clustersMap.max()) # Total number of clusters
332     #print m
333     clusterDistanceMat = numpy.zeros((m,m))
334     for i in range(m):
335       for j in range(m):
336 CId1 = i + 1
337 CId2 = j + 1
338 C1 = [[numpy.where(clustersMap == CId1)[0][k], numpy.where(
        clustersMap == CId1)[1][k]] for k in range(len(numpy.where(
        clustersMap == CId1)[0]))]
339 C2 = [[numpy.where(clustersMap == CId2)[0][l], numpy.where(
        clustersMap == CId2)[1][l]] for l in range(len(numpy.where(
        clustersMap == CId2)[0]))]
340 distanceMat = numpy.zeros((len(C1),len(C2)))
341 i2 = 0
342 for N1 in C1:
343   j2 = 0
344   for N2 in C2:
345     if CId1 != CId2:
346       distanceMat[i2,j2] = (numpy.dot( Map[N1[0],N1[1]]-Map[N2
            [0],N2[1]], numpy.transpose(Map[N1[0],N1[1]]-Map[N2
            [0],N2[1]]) ))**0.5
347     j2 = j2 + 1
348   i2 = i2 + 1
349 #print CId1, CId2
350 #print distanceMat
351 clusterDistanceMat[i,j] = distanceMat.mean()
352     self.clusterDistanceMat = clusterDistanceMat
353     return self.clusterDistanceMat
354
355   def calibration(self, Map, clustersMap, BMUs = None, name =
        False):
356     dataClusters = {}
357     if name:
358       nameClusters = {}
359     CIds = numpy.unique(clustersMap.ravel())
360     for CId in CIds:
361       dataClusters.update({CId:[]})
362       if name:
363 nameClusters.update({CId:[]})
364     for k in range(len(self.inputvectors)):
365       if BMUs == None:
366 BMU = self.findBMU(k, Map)
367       else:
368 BMU = BMUs[k]
369       CId = clustersMap[BMU[0],BMU[1]]
370       dataClusters[CId].append(self.inputvectors[k])
371       if name:
```

```
372 nameClusters[CId].append(self.inputnames[k])
373     for CId in CIds:
374       if dataClusters[CId] == []:
375 dataClusters.pop(CId)
376 if name:
377   nameClusters.pop(CId)
378     self.dataClusters = dataClusters
379     if name:
380       self.nameClusters = nameClusters
381       return self.dataClusters, self.nameClusters
382     else:
383       return self.dataClusters
384
385   def dataLooseness(self, dataClusters, clustersMap):
386     #n = len(dataClusters.keys())
387     CIds = dataClusters.keys()
388     dataLoosenessMat = numpy.ones((self.X,self.Y))*(-1)
389     for CId in CIds:
390       loosenessMat = numpy.zeros((len(dataClusters[CId]),len(
              dataClusters[CId])))
391       i = 0
392       for data1 in dataClusters[CId]:
393 j = 0
394 data1 = numpy.array(data1)
395 for data2 in dataClusters[CId]:
396   data2 = numpy.array(data2)
397   loosenessMat[i,j] = (numpy.dot( (data1-data2), numpy.
          transpose( (data1-data2) ) ))**0.5
398   j = j + 1
399 i = i + 1
400       if len(dataClusters[CId]) > 1:
401 looseness = (loosenessMat.sum()/2) / ( len(dataClusters[CId])*(
      len(dataClusters[CId]) - 1)/2 )
402       else:
403 looseness = loosenessMat.sum()/2
404       C = [[numpy.where(clustersMap == CId)[0][i], numpy.where(
              clustersMap == CId)[1][i]] for i in range(len(numpy.
              where(clustersMap == CId)[0]))]
405       for N in C:
406 dataLoosenessMat[N[0],N[1]] = looseness
407     numpy.putmask(dataLoosenessMat, dataLoosenessMat==-1, -
          dataLoosenessMat.max())
408     self.dataLoosenessMat = dataLoosenessMat
409     return self.dataLoosenessMat
410
411   def dataDistance(self, dataClusters):
412     CIds = dataClusters.keys()
413     dataDistanceMat = numpy.zeros((len(CIds),len(CIds)))
414     i = 0
```

```
415     for CId1 in CIds:
416       j = 0
417       for CId2 in CIds:
418 dataC1 = dataClusters[CId1]
419 dataC2 = dataClusters[CId2]
420 distanceMat = numpy.zeros((len(dataC1),len(dataC2)))
421 i2 = 0
422 for data1 in dataC1:
423   data1 = numpy.array(data1)
424   j2 = 0
425   for data2 in dataC2:
426     data2 = numpy.array(data2)
427     if CId1 != CId2:
428       distanceMat[i2,j2] = (numpy.dot(data1-data2, numpy.
              transpose(data1-data2)))**0.5
429     j2 = j2 + 1
430   i2 = i2 + 1
431 dataDistanceMat[i,j] = distanceMat.mean()
432 j = j + 1
433       i = i + 1
434    self.dataDistanceMat = dataDistanceMat
435    return self.dataDistanceMat
436
437  def xi(self, dataLoosenessMat, dataDistanceMat):
438    #print dataLoosenessMat
439    m = numpy.shape(dataDistanceMat)[0]
440    #print m
441    khi = sum([ e for e in numpy.unique( numpy.ravel(
          dataLoosenessMat ) ) if e >= 0])/m
442    #print khi
443    delta = (dataDistanceMat.sum() / 2) / (m*(m-1)/2)
444    #print delta
445    self.xi_value = khi / delta
446    return self.xi_value
447
448  def xiT(self, Map):
449    """
450    Find the best clustering.
451    """
452    print 'Finding the best clustering ...'
453    step = 0.01
454    Tv = numpy.arange(0,1+step,step)
455    xi_value = 9999
456    xi_old = 9999
457    BMUs = []
458    for k in range(len(self.inputvectors)):
459      BMUs.append(self.findBMU(k, Map))
460    for T in Tv:
461      clustersMap = self.clusterDiscovery(Map,T)
```

```
462         dataClusters = self.calibration(Map, clustersMap)
463         dataLoosenessMat = self.dataLooseness(dataClusters,
                clustersMap)
464         dataDistanceMat = self.dataDistance(dataClusters)
465         xi_new = self.xi(dataLoosenessMat, dataDistanceMat)
466         #print xi_new
467         if (xi_new < xi_value and xi_new != 0):
468 xi_value = xi_new
469 T_best = T
470         if xi_new > xi_old:
471 break
472         xi_old = xi_new
473         #print T
474       self.xi_best = xi_value
475       self.T_best = T_best
476       print 'Best clustering for threshold : %s' % T_best
477       return self.T_best
478
479    def tree(self, Map):
480       print 'Building tree from Kohonen map'
481       ### finding BMUs for Map
482       BMUs = []
483       for k in range(len(self.inputvectors)):
484         BMUs.append(self.findBMU(k, Map))
485       ###
486       step_value = 1E-1
487       step = step_value
488       Tv = numpy.arange(0,1+step,step)
489       clustersMap = self.clusterDiscovery(Map,0)
490       dataClusters = self.calibration(Map, clustersMap, BMUs, name
              = True)[1]
491       clusters_old = dataClusters.values() # ensemble of clusters
492       elementary_clusters = clusters_old
493       test = len(dataClusters)
494       dist = {} # distances mapping ( {[cluster]:dist} )
495       uniqueGroupTest = 0
496       #for T in Tv:
497       T = 0
498       while 1:
499         T = T + step
500         clustersMap = self.clusterDiscovery(Map,T)
501         dataClusters = self.calibration(Map, clustersMap, BMUs,
              name = True)[1]
502         clusters = dataClusters.values() # ensemble of clusters
503         keys = dataClusters.keys()
504         ngrp = len(dataClusters) # number of groups
505         #print 'ngrp=%s'%ngrp
506         #print T
507         if (ngrp == 1 and ngrp == test):
```

```
508 uniqueGroupTest = uniqueGroupTest + 1
509       if uniqueGroupTest == 1:
510 break
511       indexes = []
512       #print 'test=%s'%test
513       if (ngrp == test - 1):  # Adaptative step according to
              number of groups
514 #print T
515 #print 'ngrp == test - 1'
516 for cluster in clusters:
517   if cluster not in clusters_old:
518     for e in cluster:
519       for elementary_cluster in elementary_clusters:
520  if e in elementary_cluster:
521    indexes.append(elementary_clusters.index(elementary_cluster)
         )
522    #print elementary_cluster, elementary_clusters.index(
         elementary_cluster)
523 indexes = numpy.unique(numpy.array(indexes))
524 indexes = [int(el) for el in indexes] # to format int type and
     not numpy.int32 type
525 #print indexes
526 dist.update({tuple(indexes):T})
527 clusters_old = clusters
528 test = ngrp
529       elif ngrp == test:
530 #print 'ngrp == test'
531 step = step_value
532 test = ngrp
533       else:
534 #print 'ngrp == test - x with x > 1'
535 T = T - step
536 step = step / 10
537    #print dist
538
539    simplekeys_old = [1]
540    while 1:
541      keys = dist.keys()
542      simplekeys = [key for key in keys if len(key) == 2]
543      if simplekeys == simplekeys_old:
544 break
545       #print simplekeys
546       for key in keys:
547 keyv = list(key)
548 for simplekey in simplekeys:
549   c = 0
550   for e in simplekey:
551     #print e, key
552     #print type(e), type(key)
```

```
553         if e in key:
554           c = c + 1
555       if c==2 and len(key) > 2:
556         simple = ()
557         for e in simplekey:
558           simple = simple + (keyv.pop(keyv.index(e)),)
559         keyv.append(simple)
560     newkey = tuple(keyv)
561     value = dist.pop(key)
562     dist.update({newkey:value})
563     simplekeys_old = simplekeys
564       #print dist
565
566       keys = dist.keys()
567       #print keys
568       length = [len(key) for key in keys]
569       if max(length) > 2:
570         print 'Problem with tree building. More than two groups
              are found at the end of iterations'
571       distItems = dist.items()
572       #print distItems
573       sortList = []
574       while len(distItems) != 0:
575         maxDist = 0
576         for e in distItems:
577     if e[1] > maxDist:
578       maxDist = e[1]
579       biggestTree = e
580         sortList.append(biggestTree)
581         distItems.remove(sortList[-1])
582       #print sortList
583       #print list(sortList[0][0])
584       newick = str(sortList[0][0])
585       self.sortList = sortList[0][0]
586       #print newick
587       for i in range(len(elementary_clusters)):
588         #newick = newick.replace('%s' % i, '%s' %
              elementary_clusters[i])
589         mols = str(tuple(elementary_clusters[i])).lstrip('(').
              rstrip(')').rstrip(',')
590         #print mols
591         newick = re.sub(r'(?<=\()%s(?=,)' % i, r'(%s)'%mols,newick
              )
592         newick = re.sub(r'(?<=\b)%s(?=\))' % i, r'(%s)'%mols,
              newick)
593       #print newick
594       for subnewick in sortList:
595         subnewick = list(subnewick)
596         for i in range(len(elementary_clusters)):
```

```
597     mols = str(tuple(elementary_clusters[i])).lstrip('(').rstrip(')
            ').rstrip(',')
598     #print mols
599     subnewick[0] = re.sub(r'%s(?=,)' % i, r'(%s)'%mols, str(
            subnewick[0]))
600     subnewick[0] = re.sub(r'%s(?=\))' % i, r'(%s)'%mols, str(
            subnewick[0]))
601         newick = newick.replace('%s' % str(subnewick[0]), '%s:%s'
                % (subnewick[0], subnewick[1]) )
602         self.elementary_clusters = elementary_clusters
603         #print newick
604         newickFile = open('map.tree', 'w')
605         newickFile.write(newick)
606         newickFile.close()
607
608
609 #som = SOM()
610 #Map = som.loadMap('map_5x4.dat')
611 #Map = som.learn()
612 #som.tree(Map)
613 #som.mapPlot(Map)
614 #clusterMap = som.clusterDiscovery(Map,0.5)
615 #dataClusters = som.calibration(Map, clusterMap)
616 #dataLoosenessMat = som.dataLooseness(dataClusters, clusterMap)
617 #som.mapPlot(Map)
618 #som.mapPlot(dataLoosenessMat)
619 #clusterMap = som.clusterDiscovery(Map,0.3)
620 #print dataLoosenessMat
621 #som.mapPlot(dataLoosenessMat)
622 #dataDistanceMat = som.dataDistance(dataClusters)
623 #T_best = som.xiT(Map)
624 #clusterLoosenessMap = som.clusterLooseness(clusterMap,
        calibratedMap)
625 #som.mapPlot(clusterLoosenessMap)
626 #som.xiT(Map)
627 #clusterMap = som.clusterDiscovery(Map,0.3)
628 #print clusterLoosenessMap
629 #clusterDistanceMap = som.clusterDistance(clusterMap, Map)
630 #print som.xi(clusterLoosenessMap, clusterDistanceMap)
```

10.3 *k-means*

Implémentation python de l'algorithme *k-means*.

scripts/kmeans.py

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
```

```
3
4  from optparse import OptionParser
5  import random
6  import numpy
7  import os
8  import re
9  import progressbar
10 import cPickle
11
12 # Import Psyco if available
13 try:
14   import psyco
15   psyco.full()
16 except ImportError:
17   print 'psyco not available'
18   pass
19
20 class Kmeans:
21
22   def __init__(self, k, filename="input.dat"):
23     print '\n%s-MEANS ALGORITHM' %k
24     # ---------- READING INPUT VECTORS ----------
25     inputfile = open(filename, 'r')
26     lines = []
27     for line in inputfile:
28       lines.append(line.split())
29     self.inputvectors = [v[0:-1] for v in lines[1:]] # Input
           vectors
30     output = []
31     for v in self.inputvectors:
32       out = []
33       for e in v:
34 out.append(float(e))
35       output.append(out)
36     self.inputvectors = output
37     self.inputnames = [v[-1] for v in lines[1:]] # Vectors names
38     self.cardinal = int(lines[0][0]) # Number of element for
           each vector
39     inputfile.close()
40     # ----------
41
42     # ---------- SELECT RANDOM INITIATION CENTROIDS
           ----------
43     centroids = []
44     indexes = range(len(self.inputvectors))
45     for i in range(k):
46       #iv = random.choice(self.inputvectors)
47       index = random.choice(indexes)
48       iv = self.inputvectors[index]
```

```
49          centroids.append(iv)
50          #self.inputvectors.remove(iv)
51          indexes.remove(index)
52        #self.inputvectors.extend(centroids)
53        self.centroids = centroids
54        #print self.inputnames
55        #print self.inputvectors
56        #print self.centroids
57        #---------
58
59      def dist(self, u, v):
60        """Euclidean distance between 2 vectors"""
61        S = 0
62        for i in range(len(u)):
63          S = S + (v[i] - u[i])**2
64        dist = S**0.5
65        return dist
66
67      def clustersDiscovery(self):
68        #-------------------- INITIATION OF CLUSTER MAPPING
              ----------
69        clusters = {}
70        for i in range(len(self.centroids)):
71          clusters[i] = []
72        self.clusters = clusters
73        #---------
74        for v in self.inputvectors:
75          ds = []
76          for c in self.centroids:
77    ds.append(self.dist(v, c))
78          clid = ds.index(min(ds)) #cluster identifier
79          #print clid
80          self.clusters[clid].append(v)
81        return self.clusters # exemple of clusters dictionnary : {0:
              [[0.0, 255.0, 0.0], [20.0, 198.0, 17.0], [255.0, 255.0,
             0.0]], 1: [[0.0, 0.0, 255.0], [176.0, 20.0, 187.0], [0.0,
              0.0, 0.0], [255.0, 0.0, 0.0]], 2: [[255.0, 255.0,
             255.0]]}
82
83      def centroidsCalc(self, clusters):
84        centroids = []
85        centroidClusters = {}
86        for clid in range(len(self.centroids)):
87          #print clusters[clid]
88          centroid = []
89          for i in range(self.cardinal):
90    S = 0
91    for v in clusters[clid]:
92      S = S + v[i]
```

```
93  try:
94    m = S / len(clusters[clid])
95  except ZeroDivisionError:
96    m = 0.0
97     print "WARNING: A cluster is empty, it's crazy, isn't it !!!"
98  centroid.append(m)
99        #print centroid
100       centroids.append(centroid)
101       centroidClusters[clid] = centroid
102     #print centroids
103     self.centroids = centroids
104     self.centroidClusters = centroidClusters # exemple of
            centroidClusters : {0: [107.75, 5.0, 110.5], 1: [255.0,
            255.0, 127.5], 2: [10.0, 226.5, 8.5]}
105
106   def vectors2names(self, clusters):
107     """
108     Replace vectors by names in a clusters dictionnary
109     """
110     nameclusters = {}
111     for clid in range(len(self.centroids)):
112       #print clusters[clid]
113       cluster = clusters[clid]
114       #print self.inputvectors
115       #print self.inputnames
116       nameclusters[clid] = [self.inputnames[self.inputvectors.
              index(e)] for e in cluster]
117     self.nameclusters = nameclusters # exemple of nameclusters:
            {0: ['B', 'white'], 1: ['G', 'toto', 'Y'], 2: ['R', 'grey
            ', 'black']}
118
119   def upgma(self, kl = None): # kl: list of name of known
          ligands
120     XY = [[0,1],[0,1]]
121     print '\nUPGMA ALGORITHM FOR TREE BUILDING'
122     k = len(self.centroids)
123     newickfile = open('%s-means.tree' % k, 'w')
124     newickfile.write('('*(1+2*(k-1))) # for a k-mean we need
            1+2*(k-1) parenthesis in the newick file format
125     familier = []
126     init = True
127     initRoot = False
128     ter = False
129     # ------------------------------------ PROGRESS BAR
              ----------------------------------------------
130     widgets = ['UPGMA  ', progressbar.Percentage(), progressbar.
            Bar(marker='=',left='[',right=']')]
131     if k-3 == 0:
132       pbar = progressbar.ProgressBar(widgets=widgets, maxval=1)
```

```
133     else :
134       pbar = progressbar.ProgressBar(widgets=widgets, maxval=k
              -3+1)
135     pbar.start()
136     # ----------
137     while k>2:
138       #print k
139       newickadder = []
140       #print self.nameclusters
141       k = len(self.centroidClusters)
142       pbar.update(len(self.centroids)-k)
143       distmat = numpy.zeros((k, k))
144       for i in range(k):
145 for j in range(k):
146   distmat[i,j] = self.dist(self.centroidClusters[i], self.
          centroidClusters[j])
147       #print distmat
148       if len(distmat) != 2:
149 distmin = (distmat + numpy.ma.identity(k)*distmat.max()).min()
150 ns = (distmat + numpy.ma.identity(k)*distmat.max()).argmin(0)
151 p = (distmat + numpy.ma.identity(k)*distmat.max()).argmin()/k
152 n = ns[p]
153       elif len(distmat) == 2:
154 ter = True
155 n = 0
156 p = 1
157 distmin = distmat[n,p]
158       #print distmat[n,p]
159       if distmat[n,p] != distmin:
160 print "FATAL ERROR : problem for finding minima in distance
      matrix for UPGMA calculation"
161 os.abort()
162       #print n, p, distmin
163       #if init:
164       init1 = (str(self.nameclusters[n]).replace('[', '')).
              replace(']', '')
165       init2 = (str(self.nameclusters[p]).replace('[', '')).
              replace(']', '')
166       family = (self.nameclusters.pop(n))
167       family.extend(self.nameclusters.pop(p))
168       #print len(family)
169       for name in family:
170 if not name in familier:
171   newickadder.append(name)
172       familier.extend(family)
173       #print newickadder
174       if family == newickadder and not init:
175 #print '!!!!! NEW ROOT !!!!!'
176 #print init1
```

```
177 #print init2
178 initRoot = True
179       familyname = (str(newickadder).replace('[', '')).replace('
              ]', '')
180       if init:
181 newickfile.write('(%s),' % init1)
182 newickfile.write('(%s):%f),' % (init2,distmin))
183       elif initRoot:
184 newickfile.write( '((%s),(%s)):%f),' % (init1, init2, distmin)
      )
185       elif newickadder == []: # Ending of the initRoot
186 newickfile.write(':%f),' % distmin)
187       elif ter:
188 newickfile.write('(%s)):%f)' % (familyname,distmin) )
189       else:
190 newickfile.write('(%s)):%f),' % (familyname,distmin) )
191       ncentroid = self.centroidClusters.pop(n)
192       pcentroid = self.centroidClusters.pop(p)
193       newkey = n*10+p
194       newcentroid = []
195       for i in range(self.cardinal):
196 newcentroid.append((ncentroid[i] + pcentroid[i])/2)
197       self.centroidClusters.update({newkey: newcentroid})
198       self.nameclusters.update({newkey: family})
199       # ------------------------------ REFORMATING
              centroidClusters AND nameclusters
              ------------------------------
200       newcentroidClusters = {}
201       newnameclusters = {}
202       compt = -1
203       for key in self.centroidClusters.keys():
204 compt = compt + 1
205 newcentroidClusters.update({compt: self.centroidClusters.pop(
      key)})
206 newnameclusters.update({compt: self.nameclusters.pop(key)})
207       #print newcentroidClusters
208       #print newnameclusters
209       self.centroidClusters = newcentroidClusters
210       self.nameclusters = newnameclusters
211       #print self.centroidClusters
212       #print self.nameclusters
213       # ----------
214       init = False
215       initRoot = False
216
217        # ------------------------------ CALCULATION OF Sp AND
              Se FOR THE FAMILY ----------------------
218       if kl != None:
219 #print self.nameclusters
```

```
220     cardK = len(kl)
221     cardU = len(self.inputvectors) - cardK
222     #print cardU
223     gamma_best1 = 2**0.5
224     for clid in self.nameclusters.keys():
225       cardE = len(self.nameclusters[clid])
226       cardKaE = 0
227       for name in self.nameclusters[clid]:
228         if name in kl:
229           cardKaE = cardKaE + 1
230       cardUaE = cardE - cardKaE
231       Se = float(cardKaE) / float(cardK)
232       Sp = 1 - (float(cardUaE) / float(cardU))
233       #print Se, Sp
234       gamma = ((1-Sp)**2 + (1-Se)**2)**0.5
235       #print gamma
236       if gamma < gamma_best1:
237         gamma_best1 = gamma
238         SeBest1 = Se
239         SpBest1 = Sp
240         XY[0].append(1-Sp)
241         XY[1].append(Se)
242     #print 'best: %s %s %s'% (gamma_best1, SeBest1, SpBest1)
243     try:
244       if gamma_best1 < gamma_best2:
245         gamma_best2 = gamma_best1
246         SeBest2 = SeBest1
247         SpBest2 = SpBest1
248     except UnboundLocalError:
249       if gamma_best1 < 2**0.5:
250         gamma_best2 = gamma_best1
251         SeBest2 = SeBest1
252         SpBest2 = SpBest1
253
254       # ----------
255       pbar.finish()
256       newickfile.close()
257       # ---------------------------------------- PARENTHESIS
              EQUILIBRATION AND NEWICK FORMAT
              ----------------------------------
258       k = len(self.centroids)
259       newickfile = open('%s-means.tree' % k, 'r')
260       line = newickfile.readlines()
261       nolp = len(re.findall('\(', line[0]))
262       norp = len(re.findall('\)', line[0]))
263       diff = numpy.abs(nolp - norp)
264       newickfile.close()
265       newickfile = open('%s-means.tree' % k, 'w')
266       newickfile.write('%s' % line[0].replace(',:',':'))
```

```
267         newickfile.write(')'*diff)
268         newickfile.close()
269         if kl != None:
270           #print XY
271           rocfile = open('roc_%s-means.xy'% k, 'w')
272           cPickle.dump(XY, rocfile)
273           rocfile.close()
274           print 'best gamma, Se and Sp: %s %s %s' % (gamma_best2,
                  SeBest2, SpBest2)
275           return gamma_best2, SeBest2, SpBest2
276
277       def run(self, kl = None):
278         test = self.centroids
279         self.clustersDiscovery()
280         self.centroidsCalc(self.clusters)
281         compt = 0
282         # ------------------------------------ PROGRESS BAR
                ----------------------------------------------------
283         widgets = ['k-means    ', progressbar.Bar(marker='=',left='[',
                right=']')]
284         pbar = progressbar.ProgressBar(widgets=widgets, maxval=100)
285         pbar.start()
286         # ----------
287         while self.centroids != test:
288           test = self.centroids
289           self.clustersDiscovery()
290           self.centroidsCalc(self.clusters)
291           compt = compt + 1
292           widgets = ['%s iterations    '%compt, progressbar.Bar(marker
                  ='=',left='[',right=']')]
293           pbar = progressbar.ProgressBar(widgets=widgets, maxval=
                  compt+10)
294           pbar.update(compt)
295         pbar.finish()
296         self.vectors2names(self.clusters)
297         if kl != None:
298           gammaSeSp = self.upgma(kl)
299           return gammaSeSp
300         else:
301           self.upgma(kl)
302
303   def main(k, filename, kl = None): # kl: list of name of known
          ligands
304     if k == None and filename == None:
305       # ------------------------------ OPTION PARSER
              ----------------------------------------------
306       parser = OptionParser()
307       parser.add_option("-k", dest="k", help="number of clusters",
              metavar="Integer")
```

```
308       parser.add_option("--input", dest="input", help="Filename
              containing input vectors", metavar="input.dat")
309       (options, args) = parser.parse_args()
310       # -----------
311       kmeans = Kmeans(int(options.k), filename=options.input)
312     else:
313       kmeans = Kmeans(k, filename)
314     test = kmeans.centroids
315     kmeans.clustersDiscovery()
316     kmeans.centroidsCalc(kmeans.clusters)
317     compt = 0
318     # ------------------------------------ PROGRESS BAR
            ----------------------------------------------------
319     widgets = ['k-means  ', progressbar.Bar(marker='=',left='[',
            right=']')]
320     pbar = progressbar.ProgressBar(widgets=widgets, maxval=100)
321     pbar.start()
322     # -----------
323     while kmeans.centroids != test:
324       test = kmeans.centroids
325       kmeans.clustersDiscovery()
326       kmeans.centroidsCalc(kmeans.clusters)
327       compt = compt + 1
328       widgets = ['%s iterations  '%compt, progressbar.Bar(marker='
              =',left='[',right=']')]
329       pbar = progressbar.ProgressBar(widgets=widgets, maxval=compt
              +10)
330       pbar.update(compt)
331     pbar.finish()
332     kmeans.vectors2names(kmeans.clusters)
333     if kl != None:
334       gammaSeSp = kmeans.upgma(kl)
335       return gammaSeSp
336     else:
337       kmeans.upgma(kl)
338
339 if __name__ == "__main__":
340     main(None, None)
```

Bibliographie

[1] L.Q. Al-Mawsawi, F. Christ, R. Dayam, Z. Debyser, and N. Neamati. Inhibitory profile of a LEDGF/p75 peptide against HIV-1 integrase : insight into integrase-DNA complex formation and catalysis. *FEBS letters*, 582(10) :1425–1430, 2008.

[2] L.Q. Al-Mawsawi, V. Fikkert, R. Dayam, M. Witvrouw, T.R. Burke Jr, C.H. Borchers, and N. Neamati. Discovery of a Small-Molecule HIV-1 Integrase Inhibitor-Binding Site. *Proceedings of the National Academy of Sciences of the United States of America*, 103(26) :10080–10085, 2006.

[3] L.Q. Al-Mawsawi, A. Hombrouck, R. Dayam, Z. Debyser, and N. Neamati. Four-tiered π interaction at the dimeric interface of HIV-1 integrase critical for DNA integration and viral infectivity. *Virology*, 377(2) :355–363, 2008.

[4] J. Alvarez and B. Shoichet. *Virtual screening in drug discovery*. Taylor & Francis, 2005.

[5] M. Baba, H. Tanaka, E. De Clercq, R. Pauwels, J. Balzarini, D. Schols, H. Nakashima, C.F. Perno, RT Walker, and T. Miyasaka. Highly specific inhibition of human immunodeficiency virus type 1 by a novel 6-substituted acyclouridine derivative. *Biochemical and biophysical research communications*, 165(3) :1375–1381, 1989.

[6] H.E. Barbaree, M.C. Seto, C.M. Langton, and E.J. Peacock. Evaluating the predictive accuracy of six risk assessment instruments for adult sex offenders. *Criminal Justice and Behavior*, 28(4) :490–521, 2001.

[7] F. Barré-Sinoussi, J.C. Chermann, F. Rey, M.T. Nugeyre, S. Chamaret, J. Gruest, C. Dauguet, C. Axler-Blin, F. Vézinet-Brun, C. Rouzioux, W. Rozenbaum, and L. Montagnier. Isolation of a T-lymphotropic retrovirus from a patient at risk for acquired immune deficiency syndrome (AIDS). *Science*, 220(4599) :868–871, 1983.

[8] G. Barreiro, C.R.W. Guimarães, I. Tubert-Brohman, T.M. Lyons, J. Tirado-Rives, and W.L. Jorgensen. Search for non-nucleoside inhibi-

tors of HIV-1 reverse transcriptase using chemical similarity, molecular docking, and MM-GB/SA scoring. *Journal of Chemical Information and Modeling*, 47(6) :2416–2428, 2007.

[9] G. Bertho, J. Gharbi-Benarous, M. Delaforge, and J.P. Girault. Transferred nuclear Overhauser effect study of macrolide–ribosome interactions : correlation between antibiotic activities and bound conformations. *Bioorganic & medicinal chemistry*, 6(2) :209–221, 1998.

[10] G. Bertho, J. Gharbi-Benarous, M. Delaforge, C. Lang, A. Parent, and J.P. Girault. Conformational analysis of ketolide, conformations of RU 004 in solution and bound to bacterial ribosomes. *Journal of Medicinal Chemistry*, 41(18) :3373–3386, 1998.

[11] G. Bertho, P. Ladam, J. Gharbi-Benarous, M. Delaforge, and J.P. Girault. Conformation of macrolides antibiotics bound to ribosomes as determined from transferred nuclear Overhauser effect spectroscopy. *Journal de Chimie Physique et de Physico-Chimie Biologique*, 95(2) :423–429, 1998.

[12] G. Bertho, P. Ladam, J. Gharbi-Benarous, M. Delaforge, and J.P. Girault. Solution conformation of methylated macrolide antibiotics roxithromycin and erythromycin using NMR and molecular modelling. Ribosome-bound conformation determined by TRNOE and formation of cytochrome P450-metabolite complex. *International journal of biological macromolecules*, 22(2) :103–127, 1998.

[13] L. Berthoux, S. Sebastian, M.A. Muesing, and J. Luban. The role of lysine 186 in HIV-1 integrase multimerization. *Virology*, 364(1) :227–236, 2007.

[14] G. Bouvier, N. Evrard-Todeschi, J.P. Girault, and G. Bertho. Automatic clustering of docking poses in virtual screening process using self-organizing map. *Bioinformatics*, 26(1) :53–60, 2010.

[15] G. Bujacz, J. Alexandratos, A. Wlodawer, G. Merkel, M. Andrake, R.A. Katz, and A.M. Skalka. Binding of different divalent cations to the active site of avian sarcoma virus integrase and their effects on enzymatic activity. *Journal of Biological Chemistry*, 272(29) :18161–18168, 1997.

[16] K. Busschots, J. Vercammen, S. Emiliani, R. Benarous, Y. Engelborghs, F. Christ, and Z. Debyser. The interaction of LEDGF/p75 with integrase is lentivirus-specific and promotes DNA binding. *Journal of Biological Chemistry*, 280(18) :17841–17847, 2005.

[17] K. Busschots, A. Voet, M. De Maeyer, J.C. Rain, S. Emiliani, R. Benarous, L. Desender, Z. Debyser, and F. Christ. Identification of the

LEDGF/p75 binding site in HIV-1 integrase. *Journal of molecular biology*, 365(5) :1480–1492, 2007.

[18] H.A. Carlson, K.M. Masukawa, K. Rubins, F.D. Bushman, W.L. Jorgensen, R.D. Lins, J.M. Briggs, and J.A. McCammon. Developing a dynamic pharmacophore model for HIV-1 integrase. *Journal of Medicinal Chemistry*, 43(11) :2100–2114, 2000.

[19] M.W. Chang, R.K. Belew, K.S. Carroll, A.J. Olson, and D.S. Goodsell. Empirical entropic contributions in computational docking : Evaluation in APS reductase complexes. *Journal of Computational Chemistry*, 29(11) :1753–1761, 2008.

[20] M.W. Chang, W. Lindstrom, A.J. Olson, R.K. Belew, et al. Analysis of hiv wild-type and mutant structures via in silico docking against diverse ligand libraries. *Journal of Chemical Information and Modeling*, 47(3) :1258–1262, 2007.

[21] A. Chen and M.J. Shapiro. NOE pumping. 2. A high-throughput method to determine compounds with binding affinity to macromolecules by NMR. *Journal of the American Chemical Society*, 122(2) :414–415, 2000.

[22] P. Cherepanov. LEDGF/p 75 interacts with divergent lentiviral integrases and modulates their enzymatic activity in vitro. *Nucleic Acids Research*, 35(1) :113–124, 2007.

[23] P. Cherepanov, A.L.B. Ambrosio, S. Rahman, T. Ellenberger, and A. Engelman. Structural basis for the recognition between HIV-1 integrase and transcriptional coactivator p75. *Proceedings of the National Academy of Sciences of the United States of America*, 102(48) :17308–17313, 2005.

[24] P. Cherepanov, G. Maertens, P. Proost, B. Devreese, J. Van Beeumen, Y. Engelborghs, E. De Clercq, and Z. Debyser. HIV-1 integrase forms stable tetramers and associates with LEDGF/p75 protein in human cells. *Journal of Biological Chemistry*, 278(1) :372–381, 2003.

[25] P. Cherepanov, Z.Y.J. Sun, S. Rahman, G. Maertens, G. Wagner, and A. Engelman. Solution structure of the HIV-1 integrase-binding domain in LEDGF/p75. *Nature structural & molecular biology*, 12(6) :526–532, 2005.

[26] S.A. Chow and K.A. Vincent. Reversal of Integration and DNA Splicing Mediated by Integrase of Human Immunodeficiency Virus. *Science*, 255(5045) :723–723, 1992.

[27] A. Ciuffi, M. Llano, E. Poeschla, C. Hoffmann, J. Leipzig, P. Shinn, J.R. Ecker, and F. Bushman. A role for LEDGF/p75 in targeting HIV DNA integration. *Nature medicine*, 11(12) :1287–1289, 2005.

[28] P.R. Clapham and A. McKnight. Cell surface receptors, virus entry and tropism of primate lentiviruses. *Journal of General Virology*, 83(8) :1809–1829, 2002.

[29] R.D. Clark, A. Strizhev, J.M. Leonard, J.F. Blake, and J.B. Matthew. Consensus scoring for ligand/protein interactions. *Journal of Molecular Graphics and Modelling*, 20(4) :281–295, 2002.

[30] W.D. Cornell, P. Cieplak, C.I. Bayly, I.R. Gould, K.M. Merz, D.M. Ferguson, D.C. Spellmeyer, T. Fox, J.W. Caldwell, and P.A. Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *Journal of the American Chemical Society*, 117(19) :5179–5197, 1995.

[31] C. Dalvit, G.P. Fogliatto, A. Stewart, M. Veronesi, and B. Stockman. WaterLOGSY as a method for primary NMR screening : practical aspects and range of applicability. *Journal of Biomolecular NMR*, 21(4) :349–359, 2001.

[32] C. Dalvit, P. Pevarello, M. Tatò, M. Veronesi, A. Vulpetti, and M. Sundström. Identification of compounds with binding affinity to proteins via magnetization transfer from bulk water. *Journal of Biomolecular NMR*, 18(1) :65–68, 2000.

[33] K. Das, J. Ding, Y. Hsiou, and A.D. Clark Jr. Crystal structures of 8-Cl and 9-Cl TIBO complexed with wild-type HIV-1 RT and 8-Cl TIBO complexed with the Tyr181Cys HIV-1 RT drug-resistant mutant. *Journal of molecular biology*, 264(5) :1085–1100, 1996.

[34] E. De Clercq. Non-nucleoside reverse transcriptase inhibitors (NNRTIs) : past, present, and future. *Chemistry & biodiversity*, 1(1) :44–64, 2004.

[35] R. De J, L. Vandekerckhove, R. Gijsbers, A. Hombrouck, J. Hendrix, J. Vercammen, Y. Engelborghs, F. Christ, and Z. Debyser. Overexpression of the lens epithelium-derived growth factor/p75 integrase binding domain inhibits human immunodeficiency virus replication. *Journal of Virology*, 80(23) :11498–11509, 2006.

[36] G. De Simone, G. Balliano, P. Milla, C. Gallina, C. Giordano, C. Tarricone, M. Rizzi, M. Bolognesi, and P. Ascenzi. Human [alpha]-thrombin inhibition by the highly selective compounds N-ethoxycarbonyl–phe-pro-[alpha]-azalys p-nitrophenyl ester and N-carbobenzoxy-pro-[alpha]-

azalys p-nitrophenyl ester : A kinetic, thermodynamic and x-ray crystallographic study+, 2. *Journal of molecular biology*, 269(4) :558–569, 1997.

[37] Z. Deng, C. Chuaqui, and J. Singh. Structural Interaction Fingerprint (SIFt) : A Novel Method for Analyzing Three-Dimensional Protein- Ligand Binding Interactions. *Journal of Medicinal Chemistry*, 47(2) :337–344, 2004.

[38] P. Dorr, M. Westby, S. Dobbs, P. Griffin, B. Irvine, M. Macartney, J. Mori, G. Rickett, C. Smith-Burchnell, C. Napier, et al. Maraviroc (UK-427,857), a potent, orally bioavailable, and selective small-molecule inhibitor of chemokine receptor CCR5 with broad-spectrum anti-human immunodeficiency virus type 1 activity. *Antimicrobial agents and chemotherapy*, 49(11) :4721–4732, 2005.

[39] M.D. Eldridge, C.W. Murray, T.R. Auton, G.V. Paolini, and R.P. Mee. Empirical scoring functions : I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. *Journal of computer-aided molecular design*, 11(5) :425–445, 1997.

[40] S. Emiliani, A. Mousnier, K. Busschots, M. Maroun, B. Van Maele, D. Tempé, L. Vandekerckhove, F. Moisant, L. Ben-Slama, M. Witvrouw, F. Christ, J.-C. Rain, C. Dargemont, Z. Debyser, and R. Benarous. Integrase mutants defective for interaction with LEDGF/p75 are impaired in chromosome tethering and HIV-1 replication. *Journal of Biological Chemistry*, 280(27) :25517–25523, 2005.

[41] R. Esnouf, J. Ren, C. Ross, Y. Jones, D. Stammers, and D. Stuart. Mechanism of inhibition of HIV-1 reverse transcriptase by non-nucleoside inhibitors. *Nature Structural & Molecular Biology*, 2(4) :303–308, 1995.

[42] RM Esnouf, AL Hopkins, J. Warren, DI Stuart, and DK Stammers. Crystal structures of HIV-1 reverse transcriptase in complex with carboxanilide derivatives. *Biochemistry*, 37(41) :14394–403, 1998.

[43] N. Eswar, B. Webb, M.A. Marti-Renom, MS Madhusudhan, D. Eramian, M.Y. Shen, U. Pieper, and A. Sali. Comparative protein structure modeling using MODELLER. *Current Protocols in Protein Science*, 2(9) :1–31, 2007.

[44] T.J.A. Ewing, S. Makino, A.G. Skillman, and I.D. Kuntz. DOCK 4.0 : search strategies for automated molecular docking of flexible molecule databases. *Journal of Computer-Aided Molecular Design*, 15(5) :411–428, 2001.

[45] T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8) :861–874, 2006.

[46] T.M. Fletcher, M.A. Soares, S. McPhearson, H. Hui, M.A. Wiskerchen, M.A. Muesing, G.M. Shaw, A.D. Leavitt, J.D. Boeke, and B.H. Hahn. Complementation of integrase function in HIV-1 virions. *The EMBO Journal*, 16(16) :5123–5138, 1997.

[47] C. Flexner. HIV drug development : the next 25 years. *Nature Reviews Drug Discovery*, 6(12) :959–966, 2007.

[48] United States Food and Drug Administration. Antiretroviral drugs used in the treatment of HIV infection. http ://www.fda.gov.

[49] S. Forsén and R.A. Hoffman. Study of Moderately Rapid Chemical Exchange Reactions by Means of Nuclear Magnetic Double Resonance. *Journal of Chemical Physics*, 39 :2892–2901, 1963.

[50] C. Ganesh, A.N. Shah, CP Swaminathan, A. Surolia, and R. Varadarajan. Thermodynamic Characterization of the Reversible, Two-State Unfolding of Maltose Binding Protein, a Large Two-Domain Protein. *Biochemistry*, 36(16) :5020–5028, 1997.

[51] K. Gao, S.L. Butler, and F. Bushman. Human immunodeficiency virus type 1 integrase : arrangement of protein domains in active cDNA complexes. *The EMBO Journal*, 20(13) :3565–3576, 2001.

[52] H. Ge, Y. Si, and R.G. Roeder. Isolation of cDNAs encoding novel transcription coactivators p52 and p75 reveals an alternate regulatory mechanism of transcriptional activation. *The EMBO Journal*, 17(22) :6723–6729, 1998.

[53] J.L. Gerton and P.O. Brown. The core domain of HIV-1 integrase recognizes key features of its DNA substrates. *Journal of Biological Chemistry*, 272(41) :25809–25815, 1997.

[54] D. Ghersi and R. Sanchez. EASYMIFS and SITEHOUND : a toolkit for the identification of ligand-binding sites in protein structures. *Bioinformatics*, 25(23) :3185–3186, 2009.

[55] D. Ghersi and R. Sanchez. Improving accuracy and efficiency of blind protein-ligand docking by focusing on predicted binding sites. *Proteins : Structure, Function, and Bioinformatics*, 74(2) :417–424, 2009.

[56] F. Giordanetto, S. Cotesta, C. Catana, J.Y. Trosset, A. Vulpetti, P.F.W. Stouten, and R.T. Kroemer. Novel scoring functions comprising QXP, SASA, and protein side-chain entropy terms. *Journal of Chemical Information and Computer Science*, 44(3) :882–893, 2004.

[57] A.P. Graves, D.M. Shivakumar, S.E. Boyce, M.P. Jacobson, D.A. Case, and B.K. Shoichet. Rescoring docking hit lists for model cavity sites : predictions and experimental testing. *Journal of Molecular Biology*, 377(3) :914–934, 2008.

[58] T. Hamelryck and B. Manderick. PDB file parser and structure class implemented in Python. *Bioinformatics*, 19(17) :2308–2310, 2003.

[59] J.A. Hardy and J.A. Wells. Searching for new allosteric sites in enzymes. *Current Opinion in Structural Biology*, 14(6) :706–715, 2004.

[60] S. Hare and P. Cherepanov. The Interaction Between Lentiviral Integrase and LEDGF : Structural and Functional Insights. *Viruses*, 1(3) :780–801, 2009.

[61] S. Hare, F. Di Nunzio, A. Labeja, J. Wang, A. Engelman, and P. Cherepanov. Structural basis for functional tetramerization of lentiviral integrase. *PLoS pathogens*, 5(7) :50–264, 2009.

[62] S. Hare, S.S. Gupta, E. Valkov, A. Engelman, and P. Cherepanov. Retroviral intasome assembly and inhibition of DNA strand transfer. *Nature*, 464 :232–236, 2010.

[63] S. Hare, M.C. Shun, S.S. Gupta, E. Valkov, A. Engelman, and P. Cherepanov. A novel co-crystal structure affords the design of gain-of-function lentiviral integrase mutants in the presence of modified PSIP1/LEDGF/p75. *PLoS Pathogens*, 5(1) :e1000259, 2009.

[64] G.D. Hawkins, C.J. Cramer, and D.G. Truhlar. Pairwise solute descreening of solute charges from a dielectric medium. *Chemical Physics Letters*, 246(1-2) :122–129, 1995.

[65] Z. Hayouka, A. Levin, M. Maes, E. Hadas, D.E. Shalev, D.J. Volsky, A. Loyter, and A. Friedler. Mechanism of action of the HIV-1 integrase inhibitory peptide LEDGF 361-370. *Biochemical and Biophysical Research Communications*, 394(2) :260–265, 2010.

[66] Z. Hayouka, J. Rosenbluh, A. Levin, S. Loya, M. Lebendiker, D. Veprintsev, M. Kotler, A. Hizi, A. Loyter, and A. Friedler. Inhibiting HIV-1 Integrase by Shifting Its Oligomerization Equilibrium. *Proceedings of the National Academy of Sciences of the United States of America*, 104(20) :8316–8321, 2007.

[67] C. Hetényi and D. van der Spoel. Blind docking of drug-sized compounds to proteins with up to a thousand residues. *FEBS Letters*, 580(5) :1447–1450, 2006.

[68] D.M. Himmel, K. Das, A.D. Clark Jr, S.H. Hughes, A. Benjahad, S. Oumouch, J. Guillemont, S. Coupa, A. Poncelet, I. Csoka, et al. Crystal

structures for HIV-1 reverse transcriptase in complexes with three pyridinone derivatives : A new class of non-nucleoside inhibitors effective against a broad range of drug-resistant strains. *Journal of Medicinal Chemistry*, 48(24) :7582–7591, 2005.

[69] K. Hinsen. The molecular modeling toolkit : a new approach to molecular simulations. *Journal of Computational Chemistry*, 21(2) :79–85, 2000.

[70] J.H. Holland. *Adaptation in natural and artificial systems*. MIT press Cambridge, MA, 1992.

[71] JJ Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8) :2554–2558, 1982.

[72] A.L. Hopkins, J. Ren, R.M. Esnouf, B.E. Willcox, E.Y. Jones, C. Ross, T. Miyasaka, R.T. Walker, H. Tanaka, D.K. Stammers, et al. Complexes of HIV-1 reverse transcriptase with inhibitors of the HEPT series reveal conformational changes relevant to the design of potent non-nucleoside inhibitors. *J. Med. Chem*, 39(8) :1589–1600, 1996.

[73] N. Huang, B.K. Shoichet, and J.J. Irwin. Benchmarking sets for molecular docking. *Journal of Medicinal Chemistry*, 49(23) :6789–6801, 2006.

[74] TL Hwang and AJ Shaka. Water suppression that works. Excitation sculpting using arbitrary wave-forms and pulsed-field gradients. *Journal of Magnetic Resonance, Series A*, 112(2) :275–279, 1995.

[75] J.J. Irwin and B.K. Shoichet. ZINC- A free database of commercially available compounds for virtual screening. *J. Chem. Inf. Model*, 45(1) :177–182, 2005.

[76] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice Hall, 1988.

[77] A.N. Jain. Morphological similarity : A 3D molecular similarity method correlated with protein-ligand recognition. *Journal of Computer-Aided Molecular Design*, 14(2) :199–213, 2000.

[78] A.N. Jain. Surflex : fully automatic flexible molecular docking using a molecular similarity-based search engine. *Journal of Medicinal Chemistry*, 46(4) :499–511, 2003.

[79] A.N. Jain. Surflex-Dock 2.1 : robust performance from ligand energetic modeling, ring flexibility, and knowledge-based search. *Journal of Computer-Aided Molecular Design*, 21(5) :281–306, 2007.

[80] V. Jayalakshmi and N.R. Krishna. Complete relaxation and conformational exchange matrix (CORCEMA) analysis of intermolecular saturation transfer effects in reversibly forming ligand–receptor complexes. *Journal of Magnetic Resonance*, 155(1) :106–118, 2002.

[81] V. Jayalakshmi and N. Rama Krishna. CORCEMA refinement of the bound ligand conformation within the protein binding pocket in reversibly forming weak complexes using STD-NMR intensities. *Journal of Magnetic Resonance*, 168(1) :36–45, 2004.

[82] T.M. Jenkins, A. Engelman, R. Ghirlando, and R. Craigie. A soluble active mutant of HIV-1 integrase. *Journal of Biological Chemistry*, 271(13) :7712–7718, 1996.

[83] V.A. Johnson, F. Brun-Vézinet, B. Clotet, B. Conway, DR Kuritzkes, D. Pillay, J.M. Schapiro, A. Telenti, and DD Richman. Update of the drug resistance mutations in HIV-1 : Fall 2005. *Topics in HIV Medicine*, 13(4) :125–131, 2005.

[84] G. Jones, P. Willett, R.C. Glen, A.R. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking1. *Journal of Molecular Biology*, 267(3) :727–748, 1997.

[85] T. Kawasuji, M. Fuji, T. Yoshinaga, A. Sato, T. Fujiwara, and R. Kiyama. A platform for designing HIV integrase inhibitors. Part 2 : A two-metal binding model as a potential mechanism of HIV integrase inhibitors. *Bioorganic & medicinal chemistry*, 14(24) :8420–8429, 2006.

[86] V.V. Kharin and F.W. Zwiers. On the ROC score of probability forecasts. *Journal of Climate*, 16 :4145–4150, 2003.

[87] D.B. Kitchen, H. Decornez, J.R. Furr, and J. Bajorath. Docking and scoring in virtual screening for drug discovery : methods and applications. *Nature reviews Drug discovery*, 3(11) :935–949, 2004.

[88] A.E. Klon, M. Glick, M. Thoma, P. Acklin, and J.W. Davies. Finding more needles in the haystack : A simple and efficient method for improving high-throughput docking results. *Journal of Medicinal Chemistry*, 47(11) :2743–2749, 2004.

[89] W.C. Koff. Accelerating HIV vaccine development. *Nature*, 464(7286) :161–162, 2010.

[90] Teuvo Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences, 2001.

[91] P. Kolb, R.S. Ferreira, J.J. Irwin, and B.K. Shoichet. Docking and chemoinformatic screens for new ligands and targets. *Current opinion in biotechnology*, 2009.

[92] P.A. Kollman, I. Massova, C. Reyes, B. Kuhn, S. Huo, L. Chong, M. Lee, T. Lee, Y. Duan, W. Wang, et al. Calculating structures and free energies of complex molecules : combining molecular mechanics and continuum models. *Accounts of Chemical Research*, 33(12) :889–897, 2000.

[93] DE Koshland Jr. Application of a theory of enzyme specificity to protein synthesis. *Proceedings of the National Academy of Sciences of the United States of America*, 44(2) :98–104, 1958.

[94] I.D. Kuntz, J.M. Blaney, S.J. Oatley, R. Langridge, and T.E. Ferrin. A geometric approach to macromolecule-ligand interactions. *Journal of Molecular Biology*, 161(2) :269–288, 1982.

[95] I.D. Kuntz, J.M. Blaney, S.J. Oatley, R. Langridge, and T.E. Ferrin. A geometric approach to macromolecule-ligand interactions* 1. *Journal of Molecular Biology*, 161(2) :269–288, 1982.

[96] P.T. Lang, S.R. Brozell, S. Mukherjee, E.F. Pettersen, E.C. Meng, V. Thomas, R.C. Rizzo, D.A. Case, T.L. James, and I.D. Kuntz. DOCK 6 : Combining techniques to model RNA–small molecule complexes. *RNA*, 15(6) :1219, 2009.

[97] C. Liao and M.C. Nicklaus. Tautomerism and Magnesium Chelation of HIV-1 Integrase Inhibitors : A Theoretical Study. *ChemMedChem*, 5(7) :1053–1066, 2010.

[98] R.D. Lins, A. Adesokan, T.A. Soares, and J.M. Briggs. Investigations on human immunodeficiency virus type 1 integrase/DNA binding interactions via molecular dynamics and electrostatics calculations. *Pharmacology & Therapeutics*, 85(3) :123–131, 2000.

[99] T. Liu, Y. Lin, X. Wen, R.N. Jorissen, and M.K. Gilson. BindingDB : a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Research*, 35(Database issue) :D198–D201, 2007.

[100] M. Llano, S. Delgado, M. Vanegas, and E.M. Poeschla. Lens epithelium-derived growth factor/p75 prevents proteasomal degradation of HIV-1 integrase. *Journal of Biological Chemistry*, 279(53) :55570–55577, 2004.

[101] M. Llano, M. Vanegas, O. Fregoso, D. Saenz, S. Chung, M. Peretz, and E.M. Poeschla. LEDGF/p75 determines cellular trafficking of diverse lentiviral but not murine oncoretroviral integrase proteins and is a component of functional lentiviral preintegration complexes. *Journal of virology*, 78(17) :9524–9537, 2004.

[102] M. Llano, M. Vanegas, N. Hutchins, D. Thompson, S. Delgado, and E.M. Poeschla. Identification and characterization of the chromatin-binding domains of the HIV-1 integrase interactor LEDGF/p75. *Journal of molecular biology*, 360(4) :760–773, 2006.

[103] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2) :129–137, 1982.

[104] C. Ludwig and U.L. Guenther. Ligand based NMR methods for drug discovery. *Frontiers in Bioscience*, 14 :4565–4574, 2009.

[105] C. Ludwig, P.J.A. Michiels, A. Lodi, J. Ride, C. Bunce, and U.L. Günther. Evaluation of Solvent Accessibility Epitopes for Different Dehydrogenase Inhibitors. *ChemMedChem*, 3(9) :1371–1376, 2008.

[106] C. Ludwig, P.J.A. Michiels, X. Wu, K.L. Kavanagh, E. Pilka, A. Jansson, U. Oppermann, and U.L. Günther. SALMON : solvent accessibility, ligand binding, and mapping of ligand orientation by NMR spectroscopy. *Journal of Medicinal Chemistry*, 51(1) :1–3, 2008.

[107] P.D. Lyne. Structure-based virtual screening : an overview. *Drug Discovery Today*, 7(20) :1047–1055, 2002.

[108] G. Maertens, P. Cherepanov, W. Pluymers, K. Busschots, E. De Clercq, Z. Debyser, and Y. Engelborghs. LEDGF/p75 is essential for nuclear and chromosomal targeting of HIV-1 integrase in human cells. *Journal of Biological Chemistry*, 278(35) :33528–33539, 2003.

[109] S. Maignan, J.P. Guilloteau, Q. Zhou-Liu, C. Clément-Mella, and V. Mikol. Crystal structures of the catalytic domain of HIV-1 integrase free and complexed with its metal cofactor : high level of similarity of the active site with other viral integrases1. *Journal of molecular biology*, 282(2) :359–368, 1998.

[110] C. Marchand, A.A. Johnson, R.G. Karki, G.C.G. Pais, X. Zhang, K. Cowansage, T.A. Patel, M.C. Nicklaus, T.R. Burke, and Y. Pommier. Metal-dependent inhibition of HIV-1 integrase by β-diketo acids and resistance of the soluble double-mutant (F185K/C280S). *Molecular pharmacology*, 64(3) :600–609, 2003.

[111] G. Marcou and D. Rognan. Optimizing fragment and scaffold docking by use of molecular interaction fingerprints. *Journal of Chemical Information and Modeling*, 47(1) :195–207, 2007.

[112] R.G. Maroun, S. Gayet, M.S. Benleulmi, H. Porumb, L. Zargarian, H. Merad, H. Leh, J.F. Mouscadet, F. Troalen, and S. Fermandjian. Peptide Inhibitors of HIV-1 Integrase Dissociate the Enzyme Oligomers†. *Biochemistry*, 40(46) :13840–13848, 2001.

[113] T. Matthews, M. Salgo, M. Greenberg, J. Chung, R. DeMasi, and D. Bolognesi. Enfuvirtide : the first therapy to inhibit the entry of HIV-1 into host CD4 lymphocytes. *Nature Reviews Drug Discovery*, 3(3) :215–225, 2004.

[114] M. Mayer and B. Meyer. Characterization of ligand binding by saturation transfer difference NMR spectroscopy. *Angewandte Chemie International Edition*, 38(12) :1784–1788, 1999.

[115] M. Mayer and B. Meyer. Group epitope mapping by saturation transfer difference NMR to identify segments of a ligand in direct contact with a protein receptor. *Journal of American Chemical Society*, 123(25) :6108–6117, 2001.

[116] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4) :115–133, 1943.

[117] I.K. McDonald and J.M. Thornton. Satisfying hydrogen bonding potential in proteins. *Journal of molecular biology*, 238(5) :777–793, 1994.

[118] C.J. McKee, J.J. Kessl, N. Shkriabai, M.J. Dar, A. Engelman, and M. Kvaratskhelia. Dynamic modulation of HIV-1 integrase structure and function by cellular lens epithelium-derived growth factor (LEDGF) protein. *Journal of Biological Chemistry*, 283(46) :31802–31812, 2008.

[119] Y. Mehellou and E. De Clercq. Twenty-Six Years of Anti-HIV Drug Discovery : Where Do We Stand and Where Do We Go ? *Journal of Medicinal Chemistry*, 53(2) :521–538, 2010.

[120] EL Mehler and T. Solmajer. Electrostatic effects in proteins : comparison of dielectric and charge models. *Protein Engineering Design and Selection*, 4(8) :903–910, 1991.

[121] V.J. Merluzzi, K.D. Hargrave, M. Labadia, K. Grozinger, M. Skoog, J.C. Wu, C.K. Shih, K. Eckner, S. Hattox, J. Adams, et al. Inhibition of HIV-1 replication by a nonnucleoside reverse transcriptase inhibitor. *Science*, 250(4986) :1411–1413, 1990.

[122] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, et al. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6) :1087–1092, 1953.

[123] F. Michel, C. Crucifix, F. Granger, S. Eiler, J.F. Mouscadet, S. Korolev, J. Agapkina, R. Ziganshin, M. Gottikh, A. Nazabal, et al. Structural basis for HIV-1 DNA integration in the human genome, role of the LEDGF/P75 cofactor. *The EMBO Journal*, 28 :980–991, 2009.

[124] H. Mitsuya and S. Broder. Strategies for antiviral therapy in AIDS. *Nature*, 325 :773–778, 1987.

[125] H. Mitsuya, K.J. Weinhold, P.A. Furman, M.H. St Clair, S.N. Lehrman, R.C. Gallo, D. Bolognesi, D.W. Barry, and S. Broder. 3'-Azido-3'-deoxythymidine (BW A509U) : an antiviral agent that inhibits the infectivity and cytopathic effect of human T-lymphotropic virus type III/lymphadenopathy-associated virus in vitro. *Proceedings of the National Academy of Sciences of the United States of America*, 82(20) :7096–7100, 1985.

[126] V. Molteni, J. Greenwald, D. Rhodes, Y. Hwang, W. Kwiatkowski, F.D. Bushman, J.S. Siegel, and S. Choe. Identification of a small-molecule binding site at the dimer interface of the HIV integrase catalytic domain. *Acta Crystallographica Section D : Biological Crystallography*, 57(4) :536–544, 2001.

[127] G.M. Morris, D.S. Goodsell, R.S. Halliday, R. Huey, W.E. Hart, R.K. Belew, and A.J. Olson. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *Journal of Computational Chemistry*, 19(14) :1639–1662, 1998.

[128] D.T. Moustakas, P.T. Lang, S. Pegg, E. Pettersen, I.D. Kuntz, N. Brooijmans, and R.C. Rizzo. Development and validation of a modular, extensible docking program : DOCK 5. *Journal of Computer-Aided Molecular Design*, 20(10) :601–619, 2006.

[129] I. Muegge. Synergies of virtual screening approaches. *Mini Reviews in Medicinal Chemistry*, 8(9) :927–933, 2008.

[130] I. Muegge and Y.C. Martin. A General and Fast Scoring Function for Protein- Ligand Interactions : A Simplified Potential Approach. *Journal of Medicinal Chemistry*, 42(5) :791–804, 1999.

[131] S. Munshi, Z. Chen, Y. Li, DB Olsen, ME Fraley, RW Hungate, and LC Kuo. Rapid X-ray diffraction analysis of HIV-1 protease-inhibitor complexes : inhibitor exchange in single crystals of the bound enzyme. *Acta Crystallographica Section D : Biological Crystallography*, 54(5) :1053–1060, 1998.

[132] J. Neyman and ES Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231 :289–337, 1933.

[133] J. Neyman and ES Pearson. The testing of statistical hypotheses in relation to probabilities a priori. *Joint Statistical Papers*, pages 186–202, 1967.

[134] S.E. Nichols, R.A. Domaoal, V.V. Thakur, J. Tirado-Rives, K.S. Anderson, and W.L. Jorgensen. Discovery of wild-type and Y181C mutant non-nucleoside HIV-1 reverse transcriptase inhibitors using virtual screening with multiple protein structures. *Journal of Chemical Information and Modeling*, 49(5) :1272–1279, 2009.

[135] F.N. Novikov and G.G. Chilov. Molecular docking : theoretical background, practical applications and perspectives. *Mendeleev Communications*, 19(5) :237–242, 2009.

[136] T.E. Oliphant. Guide to NumPy. 2006.

[137] C.S. Page and P.A. Bates. Can MM-PBSA calculations predict the specificities of protein kinase inhibitors ? *Journal of Computational Chemistry*, 27(16) :1990–2007, 2006.

[138] Y. Pan, N. Huang, S. Cho, and A.D. MacKerell Jr. Consideration of molecular weight during compound selection in virtual target-based database screening. *Journal of Chemical Information and Computer Science*, 43(1) :267–272, 2003.

[139] J.D. Pata, W.G. Stirtan, S.W. Goldstein, and T.A. Steitz. Structure of HIV-1 reverse transcriptase bound to an inhibitor active against mutant reverse transcriptases resistant to other nonnucleoside inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, 101(29) :10548–10553, 2004.

[140] R. Pauwels, K. Andries, J. Desmyter, D. Schols, M.J. Kukla, H.J. Breslin, A. Raeymaeckers, J.V. Gelder, R. Woestenborghs, J. Heykants, et al. Potent and selective inhibition of HIV-1 replication in vitro by a novel series of TIBO derivatives. *Nature*, 343 :470–474, 1990.

[141] D.A. Pearlman, D.A. Case, J.W. Caldwell, W.S. Ross, T.E. Cheatham, et al. AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Computer Physics Communications*, 91(1-3) :1–41, 1995.

[142] E.F. Pettersen, T.D. Goddard, C.C. Huang, G.S. Couch, D.M. Greenblatt, E.C. Meng, and T.E. Ferrin. UCSF Chimera—a visualization system for exploratory research and analysis. *Journal of computational chemistry*, 25(13) :1605–1612, 2004.

[143] Y. Pommier, A.A. Johnson, and C. Marchand. Integrase inhibitors to treat HIV/AIDS. *Nature Reviews Drug Discovery*, 4(3) :236–248, 2005.

[144] J. Pons. *Caractérisation structurale du mode de liaison à la protéine β-TrCP, première étape vers la recherche d'inhibiteurs de NF-κB*. PhD thesis, Université Pierre et Marie Curie, 2008.

[145] J. Ren, R. Esnouf, E. Garman, D. Somers, C. Ross, I. Kirby, J. Keeling, G. Darby, Y. Jones, D. Stuart, et al. High resolution structures of HIV-1 RT from four RT-inhibitor complexes. *Nature structural biology*, 2(4) :293–302, 1995.

[146] J. Ren, R.M. Esnouf, A.L. Hopkins, E.Y. Jones, I. Kirby, J. Keeling, C.K. Ross, B.A. Larder, D.I. Stuart, and D.K. Stammers. 3'-Azido-3'-deoxythymidine drug resistance mutations in HIV-1 reverse transcriptase can induce long range conformational changes. *Proceedings of the National Academy of Sciences of the United States of America*, 95(16) :9518–9523, 1998.

[147] J. Ren, J. Milton, K.L. Weaver, S.A. Short, D.I. Stuart, and D.K. Stammers. Structural basis for the resilience of efavirenz (DMP-266) to drug resistance mutations in HIV-1 reverse transcriptase. *Structure*, 8(10) :1089–1094, 2000.

[148] J. Ren, C. Nichols, L. Bird, P. Chamberlain, K. Weaver, S. Short, DI Stuart, and DK Stammers. Structural mechanisms of drug resistance for mutations at codons 181 and 188 in HIV-1 reverse transcriptase and the improved resilience of second generation non-nucleoside inhibitors1. *Journal of molecular biology*, 312(4) :795–805, 2001.

[149] S. Renner, S. Derksen, S. Radestock, and F. Morchens. Maximum common binding modes (MCBM) : consensus docking scoring using multiple ligand information and interaction fingerprints. *Journal of Chemical Information and Modeling*, 48(2) :319–332, 2008.

[150] S. Rerks-Ngarm, P. Pitisuttithum, S. Nitayaphan, J. Kaewkungwal, J. Chiu, R. Paris, N. Premsri, C. Namwat, M. de Souza, E. Adams, et al. Vaccination with ALVAC and AIDSVAX to prevent HIV-1 infection in Thailand. *The New England journal of medicine*, 361(23) :2209–2220, 2009.

[151] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386–408, 1958.

[152] G. Rossum. Python tutorial. *CWI. Department of Computer Science [CS]*, (R 9526) :1–65, 1995.

[153] E.V. Samsonova, J.N. Kok, and A.P. IJzerman. TreeSOM : Cluster analysis in the self-organizing map. *Neural Networks*, 19(6-7) :935–949, 2006.

[154] M. Sato, T. Motomura, H. Aramaki, T. Matsuda, M. Yamashita, Y. Ito, H. Kawakami, Y. Matsuzaki, W. Watanabe, K. Yamataka, et al. Novel

HIV-1 integrase inhibitors derived from quinolone antibiotics. *Journal of Medicinal Chemistry*, 49(5) :1506–1508, 2006.

[155] A. Savarino. In-Silico docking of HIV-1 integrase inhibitors reveals a novel drug type acting on an enzyme/DNA reaction intermediate. *Retrovirology*, 4(1) :21–36, 2007.

[156] GD Schott. Penfield's homunculus : a note on cerebral cartography. *British Medical Journal*, 56(4) :329, 1993.

[157] A.R.W. Schröder, P. Shinn, H. Chen, C. Berry, J.R. Ecker, and F. Bushman. HIV-1 integration in the human genome favors active genes and local hotspots. *Cell*, 110(4) :521–529, 2002.

[158] CE Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 :1–65, 1948.

[159] M. Sippel and C.A. Sotriffer. Molecular Dynamics Simulations of the HIV-1 Integrase Dimerization Interface : Guidelines for the Design of a Novel Class of Integrase Inhibitors. *Journal of Chemical Information and Modeling*, 50(4) :604–614, 2010.

[160] P. Sneath. Numerical taxonomy. *Bergey's Manual® of Systematic Bacteriology*, pages 39–42, 1973.

[161] J. Srinivasan, T.E. Cheatham III, P. Cieplak, P.A. Kollman, and D.A. Case. Continuum Solvent Studies of the Stability of DNA, RNA, and Phosphoramidate- DNA Helices. *Journal of American Chemical Society*, 120(37) :9401–9409, 1998.

[162] M. Stahl and H.J. Böhm. Development of filter functions for protein-ligand docking. *Journal of Molecular Graphics and Modelling*, 16(3) :121–132, 1998.

[163] G. Sudlow, DJ Birkett, and DN Wade. Further characterization of specific drug binding sites on human serum albumin. *Molecular pharmacology*, 12(6) :1052–1061, 1976.

[164] J. Tang, K. Maddali, Y. Pommier, Y.Y. Sham, and Z. Wang. Scaffold Rearrangement of Dihydroxypyrimidine Inhibitors of HIV Integrase : Docking Model Revisited. *Bioorganic & Medicinal Chemistry Letters*, 20(11) :3275–3279, 2010.

[165] B. Tidor and M. Karplus. The Contribution of Vibrational Entropy to Molecular Association : : The Dimerization of Insulin. *Journal of molecular biology*, 238(3) :405–414, 1994.

[166] P. Torre III, K.J. Cruickshanks, D.M. Nondahl, and T.L. Wiley. Distortion product otoacoustic emission response characteristics in older adults. *Ear and hearing*, 24(1) :20–29, 2003.

[167] M. Totrov and R. Abagyan. Flexible ligand docking to multiple receptor conformations : a practical alternative. *Current Opinion in Structural Biology*, 18(2) :178–184, 2008.

[168] N. Triballeau, F. Acher, I. Brabet, J.P. Pin, and H.O. Bertrand. Virtual screening workflow development guided by the "receiver operating characteristic" curve approach. Application to high-throughput docking on metabotropic glutamate receptor subtype 4. *Journal of Medicinal Chemistry*, 48(7) :2534–2547, 2005.

[169] V. Tsui and D.A. Case. Theory and applications of the generalized Born solvation model in macromolecular simulations. *Biopolymers*, 56(4) :275–291, 2001.

[170] F. Turlure, G. Maertens, S. Rahman, P. Cherepanov, and A. Engelman. A tripartite DNA-binding element, comprised of the nuclear localization signal and two AT-hook motifs, mediates the association of LEDGF/p75 with chromatin in vivo. *Nucleic acids research*, 34(5) :1653–1665, 2006.

[171] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A.E. Mark, and H.J.C. Berendsen. GROMACS : fast, flexible, and free. *Journal of computational chemistry*, 26(16) :1701–1718, 2005.

[172] M.L. Verdonk, V. Berdini, M.J. Hartshorn, W.T.M. Mooij, C.W. Murray, R.D. Taylor, and P. Watson. Virtual Screening Using Protein-Ligand Docking : Avoiding Artificial Enrichment. *Journal of Chemical Information and Computer Science*, 44(3) :793–806, 2004.

[173] A.C. Wallace, R.A. Laskowski, and J.M. Thornton. LIGPLOT : a program to generate schematic diagrams of protein-ligand interactions. *Protein Engineering Design and Selection*, 8(2) :127, 1995.

[174] J. Wang, W. Wang, P.A. Kollman, and D.A. Case. Antechamber : an accessory software package for molecular mechanical calculations. *Journal of the American Chemical Society*, 222 :U403–U403, 2001.

[175] J. Wang, W. Wang, P.A. Kollman, and D.A. Case. Automatic atom type and bond type perception in molecular mechanical calculations. *Journal of Molecular Graphics and Modelling*, 25(2) :247–260, 2006.

[176] J. Wang, R.M. Wolf, J.W. Caldwell, P.A. Kollman, and D.A. Case. Development and testing of a general amber force field. *Journal of computational chemistry*, 25(9) :1157–1174, 2004.

[177] J.Y. Wang, H. Ling, W. Yang, and R. Craigie. Structure of a two-domain fragment of HIV-1 integrase : implications for domain organization in the intact protein. *The EMBO Journal*, 20(24) :7333–7343, 2001.

[178] W. Wang and P.A. Kollman. Computational study of protein specificity : The molecular basis of HIV-1 protease drug resistance. *Proceedings of the National Academy of Sciences of the United States of America*, 98(26) :14937–14942, 2001.

[179] G.L. Warren, C.W. Andrews, A.M. Capelli, B. Clarke, J. LaLonde, M.H. Lambert, M. Lindvall, N. Nevins, S.F. Semus, S. Senger, et al. A critical assessment of docking programs and scoring functions. *Journal of Medicinal Chemistry*, 49(20) :5912–5931, 2006.

[180] J. Wielens, S.J. Headey, D. Jeevarajah, D.I. Rhodes, J. Deadman, D.K. Chalmers, M.J. Scanlon, and M.W. Parker. Crystal structure of the HIV-1 integrase core domain in complex with sucrose reveals details of an allosteric inhibitory binding site. *FEBS letters*, 584(8) :1455–1462, 2010.

[181] R. Zheng, T.M. Jenkins, and R. Craigie. Zinc folds the N-terminal domain of HIV-1 integrase, promotes multimerization, and enhances catalytic activity. *Proceedings of the National Academy of Sciences of the United States of America*, 93(24) :13659–13664, 1996.

[182] Y. Zheng, Z. Ao, J.K. Danappa, and X. Yao. Characterization of the HIV-1 integrase chromatin-and LEDGF/p75-binding abilities by mutagenic analysis within the catalytic core domain of integrase. *Virology journal*, 7(1) :68–81, 2010.

[183] M.H. Zweig and G. Campbell. Receiver-operating characteristic (ROC) plots : a fundamental evaluation tool in clinical medicine. *Clinical Chemistry*, 39(4) :561–577, 1993.

Index