



École Doctorale SPI

Laboratoire CRISTAL

Thèse

Présentée pour l'obtention du grade de DOCTEUR
DE L'UNIVERSITE DE LILLE

par

Yoann Dufresne

Algorithmique pour l'annotation automatique de peptides non ribosomiques

Spécialité : Informatique, Bioinformatique

Soutenue le 1 Décembre 2016 devant un jury composé de :

Rapportrice	Marie-France Sagot	(TODO)
Rapportrice	Frédérique Lisacek	(Groupe d'Informatique Proteomique, SIB Genève)
Examinateur	Pablo Carbonell	(SYNBIOCHEM, University of Manchester)
Examinatrice	Sophie Tison	(Laboratoire CRISTAL, univ Lille 1)
Directrice de thèse	Maude Pupin	(Laboratoire CRISTAL, univ Lille 1)
Co-encadrant de thèse	Laurent Noé	(Laboratoire CRISTAL, univ Lille 1)
Invitée	Valérie Leclère	(Institut Charles Viollette, univ Lille 1)

Affiliations :

1 -

2 - SIB Institute Suisse de Bioinformatique, Groupe d'Informatique Proteomique, Genève
3 - BBSRC/EPSRC Synthetic Biology Research Centre for Fine and Speciality Chemicals
(SYNBIOCHEM), Manchester Institute of Biotechnology, Faculty of Science and Engineering,
University of Manchester, 131 Princess Street Manchester M1 7DN, UK

4 -



Thèse effectuée au sein du **Laboratoire CRISTAL**

de l'Université de Lille
Bâtiment M3 extension
avenue Carl Gauss
59655 Villeneuve d'Ascq Cedex
France

Résumé

Rédigez votre résumé en français ici.

Mots-clé:

Quelques mots-clé.

Title of your thesis in English

Abstract

You should write your abstract in English here.

Keywords:

Some keywords.

Remerciements

Remerciements en français ou Acknowledgements en anglais. Ceci devraient être écrits après la soutenance.

Contents

Introduction	1
Prérequis	3
0.1 Représentation 2D de molécules	3
0.2 Représentation 1D d'une molécule : les SMILES	4
0.3 Graphes	5
1 Les peptides non ribosomiques	7
1.1 Synthèse non-ribosomique	7
1.1.1 Introduction	7
1.1.2 Généralités sur les peptides non ribosomiques	8
1.1.3 Généralités sur les synthétases	14
1.1.4 Les domaines principaux	15
1.1.5 Les domaines de modification des monomères	18
1.1.6 Incorporations extra-NRPS	20
1.2 Les outils bioinformatiques pour l'annotation et l'analyse des NRP/NRPS .	24
1.2.1 Les outils d'annotation de NRPS	24
1.2.2 Les bases de connaissances de NRPS	27
1.2.3 L'annotation de NRP	28
1.2.4 Les bases de connaissance de NRP	31
2 s2m : Des atomes vers les monomères	33
2.1 Introduction	33

2.2	Formalisation du problème d'annotation	34
2.2.1	Définition du problème	34
2.2.2	L'existant	35
2.2.3	Vers des problèmes informatiques	40
2.3	Sous-graphe Maximum Commun vs Isomorphisme de Sous-graphe	42
2.3.1	Sous-graphe Maximum Commun	43
2.3.2	Isomorphisme de sous-graphe	47
2.3.3	Choisir l'algorithme de recherche de monomères	52
2.4	Construction de Smiles2Monomers	52
2.4.1	Isomorphisme de sous-graphe appliqué à la recherche de monomères	52
2.4.2	Des monomères aux résidus	64
2.4.3	Pavage de monomères	67
2.4.4	Recherche approximative (light)	76
2.4.5	Vue globale des algorithmes	77
2.5	Résultats et interprétations	79
2.5.1	Jeux de données	79
2.5.2	Profil d'un résultat	80
2.5.3	Choix des paramètres	82
2.5.4	Répartition des temps de calcul et analyse	88
2.5.5	Analyse des résultats	90
3	Vers un enrichissement de Norine	97
3.1	Norine	97
3.1.1	Généralités	97
3.1.2	Les peptides	98
3.1.3	Les outils liés	101
3.2	Les contributions de s2m à Norine	103
3.2.1	Améliorations de l'existant	104
3.2.2	Mises à jour de Norine	108
3.3	Vers la biologie de synthèse	114

3.3.1	L'existant	114
3.3.2	Aider les techniques de recombinaisons modulaires	117
Conclusions et perspectives		125
Bibliography		140

Préambule

Dans ce manuscrit, vous trouverez l'ensemble des résultats de mes travaux de thèse. Avant de les présenter, je vais ici introduire le contexte dans lequel j'ai effectué ces travaux. Je suis issu d'une formation en informatique fondamentale de Lille 1 et spécialisé en algorithmique par un master de cette même université. C'est dans le cadre de mon parcours universitaire que j'ai eu l'occasion de rencontrer Maude Pupin, mon actuelle directrice de thèse, et Laurent Noé co-encadrant. Ils m'ont accueilli plusieurs fois en projets de master et stages, puis en thèse au sein de l'équipe Bonsai dirigée par Hélène Touzet, équipe de bioinformatique commune au laboratoire CRISTAL et INRIA.

L'équipe Bonsai est une équipe de bioinformatique orientée algorithmique. Plus précisément, l'équipe est orientée algorithmique des séquences, qu'elles soient ARN, ADN, de gènes ou moléculaires. Ma thématique de recherche est elle orientée structures moléculaire pour des molécules appelées peptides non ribosomiques. Au cours de mes 3 années de thèse j'ai interagi à de nombreuses reprises avec d'autres thématiques de l'équipe, mais deux discussions ont particulièrement menées à des collaborations hors de mon cadre de recherche usuel.

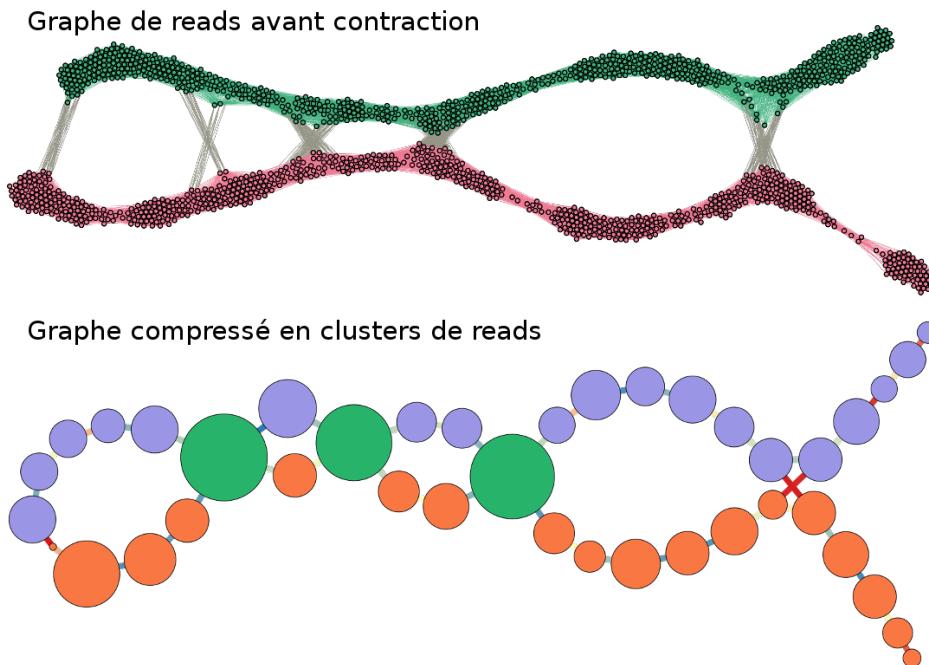


Figure 1: Contraction d'un graphe de read pour extraire les clusters à assembler ensuite. Ces données sont issues d'un métagénome simple à deux espèces.

Pierre Péricard est également doctorant de l'équipe et travaille sur un pipeline d'as-

semblage de marqueurs conservés dans des données métagénomiques. Durant le déroulement algorithmique du programme, il est amené à construire un graphe de reads présents dans l'échantillon où deux reads sont liés lorsqu'ils sont partiellement chevauchants (voir figure 1). Il souhaitait pouvoir extraire les composantes "linéaires" pour effectuer des assemblages sur ces sous-jeux de read. J'ai créé pour lui un programme qui permet de réduire ce graphe avec une épaisseur en un graphe filiforme facilement découpable au niveau des arêtes faibles (en rouge sur la figure) et des noeuds d'arité supérieure à 2. Cet utilitaire permet la contraction des noeuds proches via des distances calculées par l'algorithme de Dijkstra, ainsi que l'utilisation de plusieurs filtres discriminant les reads ne contribuant pas à l'homogénéité du graphe. Le pipeline complet sera prochainement publié par Pierre.

En début d'année, Hélène Touzet a publié un article contenant une méthode de programmation dynamique permettant le comptage des voisins d'un mot avec k erreurs. Cette méthode utilise le croisement de deux automates dont l'un est l'automate universel déterministe de Levenshtein. Le fait que cet automate universel ne dépende pas du mot pour lequel on recherche le voisinage, permet de le générer une seule et unique fois pour un k donné, pour n'avoir plus qu'à le croiser avec le second automate lors de chaque exécution. Depuis cette publication, je me suis proposé pour travailler avec Hélène sur l'implémentation de la méthode puis sur l'analyse de la minimalité de l'automate universel déterministe de Levenshtein que nous avons générée. Ces travaux sont en cours et feront l'objet du publication prochaine.

Mon sujet de thèse m'a également amené à collaborer avec de nombreuses personnes locales mais également d'équipes distantes. Localement, j'ai travaillé avec plusieurs membres de l'institut Charles Violette, institut également hébergé par l'université Lille 1. J'ai notamment beaucoup travaillé avec Valérie Leclère sur les aspects de biologie de mon sujet et Mickael Chevalier concernant les aspects de spectrométrie de masse.

À l'extérieur, j'ai principalement travaillé avec Tilmann Weber et son équipe hébergée au Danemark par la Novo Nordisk Foundation. Ses travaux sur l'outil Antismash en font un collaborateur de longue durée au sein de la thématique des peptides non ribosomiques sur l'université Lille 1. C'est d'ailleurs cette collaboration entre son équipe et la notre qui a permis l'organisation, à Lille et par deux fois (en 2013 et 2015), d'un workshop sur la thématique des outils bioinformatiques pour l'analyse des peptides non ribosomiques et des polykétides. Cette collaboration de longue date m'a également permis d'être accueilli durant un mois au Danemark dans son équipe pour démarrer avec son équipe, le développement d'une nouvelle application dont nous parlerons dans la partie 3.3 de ce manuscrit.

Depuis maintenant plus d'un an, l'équipe autour de Norine collabore également avec l'équipe de recherche Suisse "Proteome Informatics" dirigée par Frédérique Lisacek, dans le but de créer un pipeline d'identification de NRP via spectrométrie de masse. Deux doctorants ont été recrutés à Lille (Mickael Chevalier) et Genève (Emma Ricart). Durant ma thèse, j'ai collaboré avec Emma pour son utilisation de smiles2monomers (logiciel développé durant ma thèse) afin de valider les structures biologiques de ses peptides non ribosomiques.

Globalement, mon contexte de travail a été particulièrement agréable et m'a permis de m'ouvrir l'esprit à la recherche mais également à la vie universitaire en générale. Les 3 ans et 6 mois d'enseignement en informatique et les différentes responsabilités universitaires, telles que la création et gestion d'un groupe d'entraînement algorithmique avec Thibault Raffaillac, m'ont préparé à l'éventualité de devenir un jour maître de conférence ou chercheur.

Introduction

Les peptides non ribosomiques (NRP en anglais) sont des molécules synthétisées par des bactéries et champignons microscopiques. Ces molécules sont d'une importance capitale pour ces organismes car elles sont souvent utilisées comme mécanisme de défense contre d'autres micro-organismes. Nous, humains, nous intéressons à ces molécules car elles sont une source importante de nouvelles molécules pour la pharmacologie. En particulier, une très grande partie des NRP découverts à ce jour ont des propriétés antibiotiques. La célèbre pénicilline, découverte par Alexander Fleming au début du siècle dernier, est une molécule issue d'une transformation d'un précurseur NRP.

Comme leur nom l'indique, les peptides non ribosomiques ne sont pas produits par la voie de synthèse classique des protéines (utilisant le ribosome). Cette voie de synthèse alternative autorise la cellule à créer des molécules de formes et de compositions inhabituelles. Les molécules sont assemblées à partir d'éléments de base appelés monomères. Il existe actuellement plus de 500 monomères différents répertoriés comme étant inclus au sein de NRP. Ce sont les compositions inhabituelles couplées aux formes particulières qui confèrent leur diversité d'activité et leur efficacité aux NRP. Connaître les compositions des NRP est d'une importance cruciale car c'est cela qui nous permet de relier un composé à sa voie de synthèse et aussi de prédire l'activité que peut avoir cette molécule.

Les structures NRP sont découvertes via deux méthodes. D'un côté, il existe des logiciels d'analyse d'ADN qui détectent les gènes menant à la création de ces molécules. Ces logiciels prédisent, à partir de l'ADN, les différents constituants potentiels des NRP produits. Cependant, en l'état actuel des connaissances, ces techniques ne peuvent pas complètement inférer les compositions et formes complètes des peptides qui seront synthétisés. D'un autre côté, il est possible de découvrir expérimentalement des NRP en analysant les composés produits par les organismes. Ce processus permet d'obtenir, par spectrométrie de masse entre autres, les structures chimiques des molécules. Cependant, pour obtenir les informations biologiques (les monomères présents dans le peptide), il est souvent nécessaire d'effectuer une annotation manuelle. Là où la première méthode de découverte donne

rapidement de nombreuses annotations incomplètes, la seconde méthode donne des annotations exactes mais avec un bien plus faible débit du fait du traitement manuel. Ma thèse s'articule autour de l'obtention rapide et exacte des annotations biologiques et de leur utilisation.

Le manuscrit sera découpé en trois chapitres. Le premier chapitre abordera les notions de biologie nécessaires à la compréhension de la suite et sera découpé en deux sections. La section 1.1 abordera dans le détail les voies de synthèse des peptides non ribosomiques et la section 1.2 donnera une vue globale des différents logiciels de découverte et stockage d'annotations des NRP.

Le second chapitre abordera la génération automatique d'annotations biologiques de NRP et sera découpé en 5 sections. Après une courte introduction, la section 2.2 définira précisément la problématique informatique de graphes liée à l'annotation automatique. Elle sera suivie en section 2.3 par un état de l'art de solutions algorithmiques pour résoudre ces problèmes. Puis au sein de la section 2.4 nous expliquerons en détail notre construction algorithmique pour produire rapidement et automatiquement des annotations biologiques. Enfin, en section 2.5, nous effectuerons de nombreux tests pour valider le modèle algorithmique proposé précédemment.

Le dernier chapitre, découpé en 3 sections sera, quant à lui, consacré à l'utilisation des résultats produits par le logiciel d'annotation. La section 3.1 détaillera le fonctionnement de la base de donnée Norine, base au sein de laquelle nous avons intégré les annotations générées. Puis nous expliquerons en détail en section 3.2, comment notre logiciel a contribué à l'amélioration de la qualité et de la quantité d'informations dans Norine. Enfin, en section 3.3, nous aborderons les perspectives d'utilisation des annotations biologiques pour avancer progressivement vers la création de NRP de synthèse.

Prérequis

Pour comprendre le manuscrit dans son intégralité, il sera nécessaire de comprendre quelques outils. Ces outils n'entrant pas dans le récit de la thèse, j'ai choisi de les décrire dans cette section de prérequis. Je vais donc présenter ici les représentations en 2 dimensions et en chaînes de caractères des molécules ainsi que l'outil informatique des graphes que nous allons utiliser pour représenter nos données.

0.1 Représentation 2D de molécules

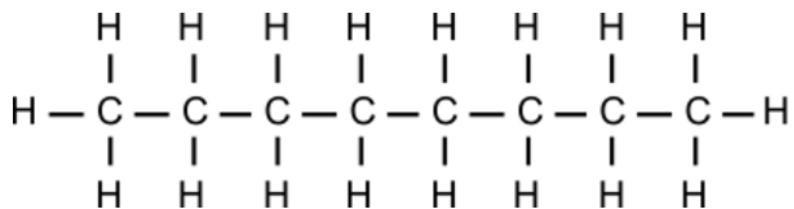


Figure 2: Formule développée de la molécule d'octane.

L'ensemble du manuscrit est orienté vers l'étude de molécules issues de constructions biologiques. Une molécule peut classiquement être représentée en 2D sous la forme d'un dessin d'atomes représentés par des lettres, liés par des traits représentant les liaisons covalentes entre atomes (voir figure 2). Cette représentation est appelée *formule développée* de la molécule.

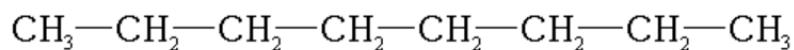


Figure 3: Formule semi-développée de la molécule d'octane.

Les atomes d'hydrogène étant omniprésents, la représentation de grandes molécules peut aboutir sur une formule développée volumineuse. Pour simplifier leur représenta-

tion a été inventée la forme dite *semi-développée* (voir figure 3). Puisque les atomes d'hydrogène n'effectuent qu'une seule liaison, la représentation de ceux-ci peut être contracté avec l'atome qu'ils accompagnent, sans pour autant laisser un doute sur les liaisons non représentées.

0.2 Représentation 1D d'une molécule : les SMILES

Les librairies de chemoinformatique utilisent régulièrement un format plus compact pour les représentations moléculaires. Toute la structure est alors compactée en un texte. Ce format est appelé SMILES (pour *Simplified Molecular Input Line Entry Specification*) et a été publié en 1988 [?]. L'octane représenté en figures 2 et 3 peut par exemple être écrit sous la forme verbeuse du SMILES :

[C]([H])([H])([H])[C]([H])([H])[C]([H])([H])[C]([H])([H])[C]([H])([H])[C]([H])([H])
[C]([H])([H])[C]([H])([H])[H]

Chaque atome est ici représenté par une lettre entourée de crochets. Toute sous partie de l'expression mise entre parenthèses représente un embranchement à partir de l'atome précédent. Sur cet exemple, la chaîne principale est donc constituée de [C][C][C][C][C][C][C][C][H]. Chacun des [C] est accompagné de plusieurs branchements ne contenant qu'un seul hydrogène. Par exemple le début de la chaîne est constituée d'un [C]([H])([H])([H]), ce qui veut dire que le premier carbone, en plus d'être lié au carbone suivant, possède trois branches constituées d'un unique hydrogène. Il est à noter qu'il existe de nombreux SMILES pour représenter cette molécule.

J'ai arbitrairement choisi de commencer par le carbone le plus à gauche de la figure 3, mais j'aurais aussi bien pu démarrer de n'importe quel atome plus au centre.

La notation présentée est lourde mais deux règles supplémentaires viennent réduire la notation. Premièrement, les atomes d'hydrogène sont facultatifs. Même si ils ne sont pas présents dans l'écriture, ils seront automatiquement ajoutés jusqu'à saturation de la valence des atomes hôtes. Deuxièmement, il est autorisé de ne pas mettre les crochets pour les atomes les plus fréquemment utilisés en chimie organique (par exemple C, N ou O). Avec ces deux assouplissements, notre molécule d'octane s'écrit donc de la manière suivante : CCCCCCCC.

Les SMILES doivent inclure plusieurs autres règles pour pouvoir représenter l'intégralité des molécules possibles. Ainsi, un = ou un # sont ajoutés entre deux atomes pour représenter des liaisons doubles ou triples. Par exemple, le SMILES C(=O)OH représente un groupement carboxyle. Le =O est une branche, contenant un unique atome d'oxygène, reliée au carbone par une double liaison.

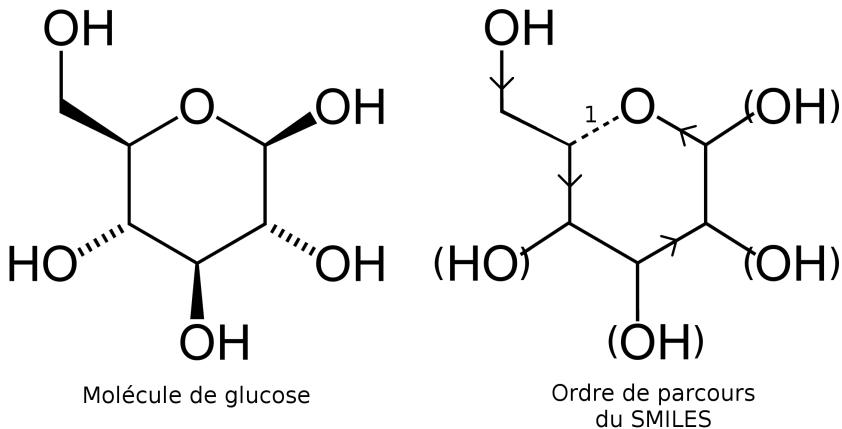


Figure 4: Molécule de glucose et son ordre de parcours dans le SMILES cyclique exemple.

Les molécules peuvent également être cycliques. Il n'est pas possible de représenter un cycle dans un texte sans ajouter une notation supplémentaire. Ainsi, on ajoute un nombre à la suite de chacun des deux atomes distants dans la chaîne de caractère, mais reliés dans la molécule. La molécule de glucose représentée sur la figure 4 s'écrira donc OCC1C(O)C(O)C(O)C(O)O1. La valeur 1 permet de représenter la liaison entre l'atome d'oxygène dans le cycle et le carbone juste à sa gauche.

Je n'ai décrit ici que les règles qui nous seront utiles. L'intégralité des spécifications est disponible en ligne sur de nombreux sites.

0.3 Graphes

Un graphe est un outil permettant de représenter des données sous la forme de nœuds reliés par des arcs lorsque ceux-ci sont orientés ou par des arêtes pour les non orientés (exemple sur la figure 5). Les noeuds des graphes peuvent être étiquetés ou non. Sur l'exemple chaque étiquette de nœud est représentée par une couleur. Le nœud 1 est étiqueté par "bleu", les nœuds 2, 3 et 5 par "vert" et enfin le nœud 4 par "rouge". Dans la suite du manuscrit, nous utiliserons souvent cette représentation colorée des étiquettes pour plus de clarté.

Cette représentation de données est très souvent utilisée en informatique et ce, pour

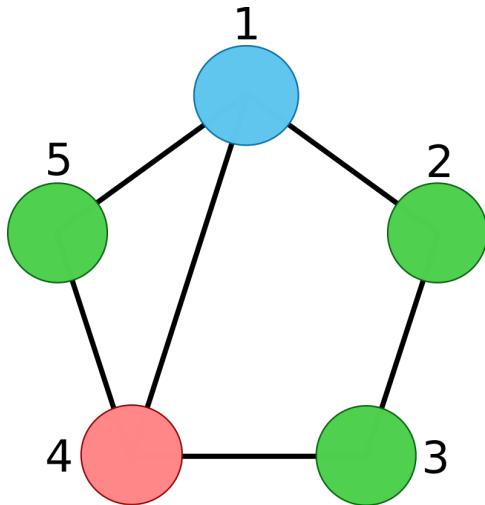


Figure 5: Exemple de graphe non orienté contenant 5 noeuds.

tout type de domaines. Dans notre cas nous allons l'utiliser à de multiples reprises, et en particulier pour représenter les molécules sous leur forme atomique ainsi que sous leur forme biologique. Nous verrons dans le second chapitre qu'il est possible de transformer nos molécules en graphes atomiques non orientés dont chacun des nœuds est étiqueté avec des informations sur les atomes. Cette représentation nous permettra d'appliquer des algorithmes classiques à nos données particulières.

Chapter 1

Les peptides non ribosomiques

1.1 Synthèse non-ribosomique

1.1.1 Introduction

Afin de bien comprendre la voie de synthèse non ribosomique, commençons par quelques rappels rapides sur la synthèse des protéines classiques. Dans la cellule, les protéines sont assemblées par un complexe moléculaire, appelé ribosome, qui lit les ARN messagers. Ces ARN sont les vecteurs de l'information génétique et sont issus de la transcription d'un morceau d'ADN. Ils sont traduits par triplets de nucléotides (appelés codons) en chaînes d'acides aminés. Les 64 codons possibles (4^3 nucléotides) sont pour la plupart traduits en 20 acides aminés appelés acides aminés protéogéniques (car ils interviennent dans la synthèse classique de protéines). Ces acides aminés sont tous composés d'un même squelette atomique autorisant deux liaisons et permettant ainsi la formation de chaînes peptidiques. Sur le squelette est ancrée une chaîne latérale variant d'un acide aminé à l'autre, leur donnant leur spécificité (voir figure 1.1). Les deux liaisons qu'effectue le squelette sont supportées par un groupement amine (NH_2) et un groupement carboxyle ($C(=O)OH$). Ces deux groupements se lient entre eux et créent ainsi une protéine linéaire. Lorsque la chaîne n'est constitué que de quelques acides aminés (généralement moins de 25) on ne la nomme plus protéine mais peptide.

Une fois assemblée, une protéine se replie sur elle même et les caractéristiques structurelles et physico-chimiques qui en découlent lui donnent son activité. Plus précisément, les propriétés des éléments en contact avec d'autres molécules détermineront l'activité de cette protéine. Il est possible que ces surfaces soient à "l'extérieur" de la protéine ou à "l'intérieur" sous forme de poche. Ces propriétés sont donc très dépendantes du repliement et

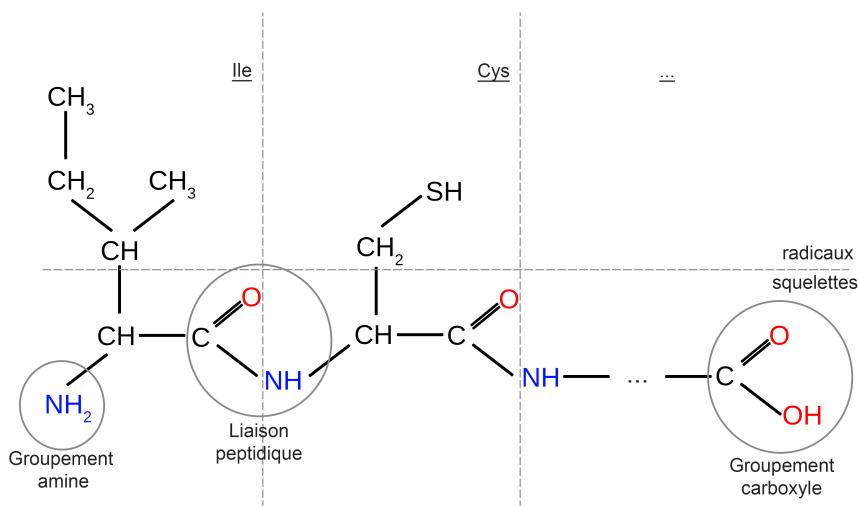


Figure 1.1: Présentation d'une chaîne peptidique. Chaque trait pointillé vertical sépare un acide aminé d'un autre et le trait horizontal sépare le squelette de la chaîne latérale des acides aminés.

des types des acides aminés exposés.

1.1.2 Généralités sur les peptides non ribosomiques

Les peptides non ribosomiques (Non Ribosomal Peptide -NRP-) sont des petits polymères synthétisés par certaines bactéries et certains champignons unicellulaires. Tout comme les protéines classiques, les NRP sont des molécules résultant d'assemblages de briques de base. Cependant, comme le nom l'indique, la voie de synthèse d'un NRP est différente de celle d'une protéine classique. Cette voie de synthèse comporte une étape supplémentaire (voir figure 1.2). Comme nous l'avons présenté en 1.1.1, lors d'une création classique de protéine, l'ADN est transcrit en ARN qui lui-même est traduit en protéine. Dans le cas d'une NRPS, la protéine produite n'est pas le produit final mais une *enzyme modulaire* agissant seule ou en complexe afin d'assembler les NRP. Ces complexes sont appelés des synthétases (Non Ribosomal Peptide Synthetase -NRPS-).

1.1.2.1 Les monomères

Tandis que les protéines classiques sont majoritairement composées des 20 acides aminés standards, la synthèse non ribosomique incorpore plusieurs centaines de briques de base différentes. Ces briques de base sont appelées *monomères*. La base de données de référence

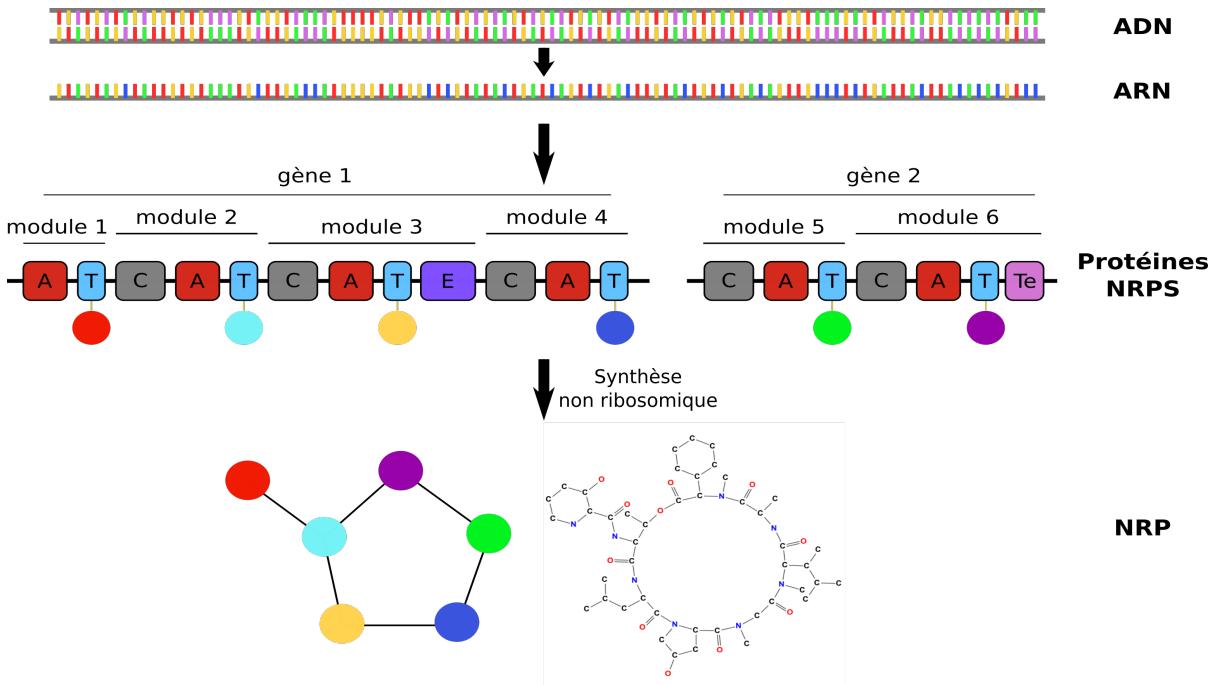


Figure 1.2: Voie de synthèse non ribosomique : groupés en cluster, les gènes codant les protéines NRPS sont transcrits puis traduits par le ribosome. Les domaines A de la NRPS capturent ensuite des monomères (billes de couleur ici) dans l'environnement afin de les assembler en peptide. Enfin, le NRP assemblé est relâché.

des NRP compte pour le moment 533 monomères différents. Les monomères peuvent provenir de différents groupes. Parmi ces monomères, on compte les 20 acides aminés standards ainsi qu'un grand nombre de dérivés proches. Après plusieurs modifications que nous détaillerons en section 1.1.5, il est par exemple possible d'obtenir des monomères méthylés ou oxydés (voir figure 1.3). On peut également citer les sucres et les acides gras comme faisant partie des monomères candidats à l'inclusion dans des NRP.

1.1.2.2 Les structures peptidiques

Contrairement à la création des peptides classiques par le ribosome, les NRPS peuvent assembler les monomères au sein des NRP de manière non linéaire. Certains monomères possèdent plus de deux groupements capables de réagir et former une liaison avec un autre monomère. Ainsi sur l'exemple de la figure 1.4, on peut constater que le monomère nommé Dpr (acide 2,3-diamonopropionique) possède un groupement amine supplémentaire à celui déjà présent dans le squelette des acides aminés classiques. Ce groupement en bout de chaîne latérale autorise le Dpr à se lier 3 fois et ainsi casser la linéarité de la molécule as-

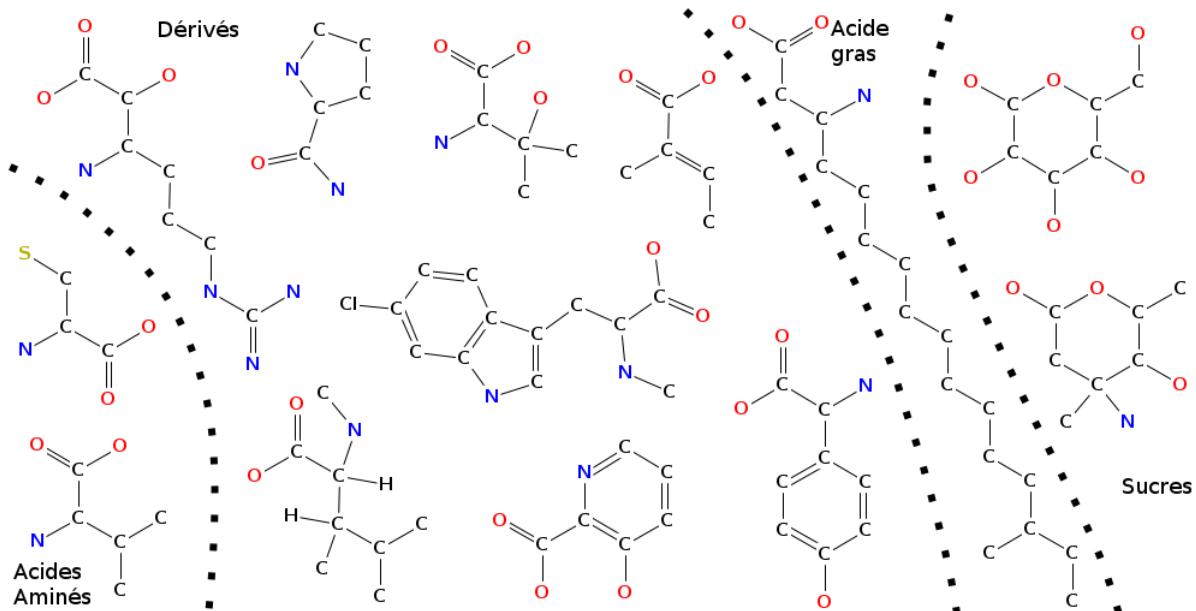


Figure 1.3: Patchwork de monomères NRP. De gauche à droite : deux représentants des acides aminés classiques ; plusieurs dérivés d'acides aminés ; un acides gras ; deux sucres.

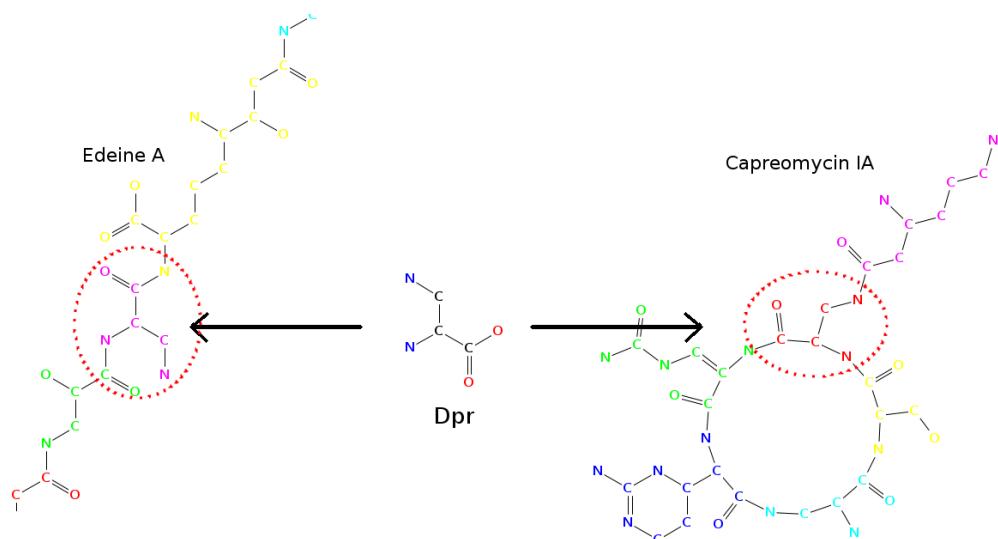


Figure 1.4: Deux exemples d'inclusion du monomère Dpr : Au sein de l'Edeine A, seules les deux liaisons classiques du squelette peptidique sont effectuées entre le Dpr (en rose) et les autres monomères. Au sein de la Capreomycin IA, une troisième liaison est effectuée par le Dpr (en rouge) depuis son groupement amine (NH_2) de la chaîne latérale.

semblée. Ces monomères permettent ainsi d'obtenir des structures à embranchements.

1.1.2.3 Les liaisons inter-monomères

Classiquement, les liaisons au sein de peptides se font par le rapprochement d'un groupement carboxyle d'un premier monomère vers le groupement amine d'un second. Au sein des NRP, plusieurs autres types de liaisons viennent s'ajouter aux liaisons peptidiques. Ceci augmente la diversité structurale des peptides. La liaison peptidique classique reste tout de même la principale liaison effectuée entre monomères. En dehors de celle-ci, nous pouvons lister trois différentes façons de lier les NRP.

Le premier type de liaison inclut un monomère contenant un atome de soufre. Comme pour les protéines classiques, les monomères soufrés peuvent effectuer des ponts disulfure (liaison entre deux atomes de soufre en perdant deux atomes d'hydrogène). Cependant, ce type de liaison n'est pas le seul impliquant un soufre. Comme nous le verrons plus tard lors de la description des modules NRPS Cy (voir 1.1.4.3), l'atome de soufre peut également intervenir dans une cyclisation entre deux monomères en perdant son atome d'hydrogène de la même manière que lorsqu'il réalise un pont disulfure (voir figure 1.5).

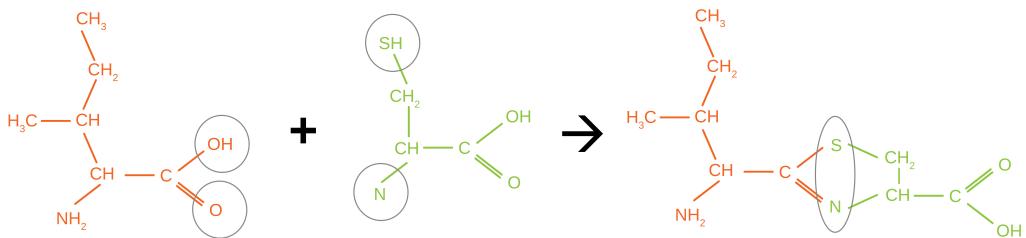


Figure 1.5: Exemple de liaison effectuée par un domaine Cy

Le second type de liaison implique au moins un monomère contenant un cycle aromatique (cycle imidazole ou benzène). Dans les deux cas, c'est un atome d'hydrogène qui est arraché au cycle afin de pouvoir y lier un monomère. Lors d'une liaison sur un cycle imidazole, c'est l'un des deux carbones voisins dans le cycle qui perdra un de ses hydrogènes au profit de la liaison avec l'autre monomère. Dans le cas d'un benzène (Voir figure 1.6), c'est la présence d'un oxygène accroché au cycle qui permettra une affinité avec les carbones voisins.

Enfin, le dernier type de liaison que nous allons décrire implique au moins un monomère

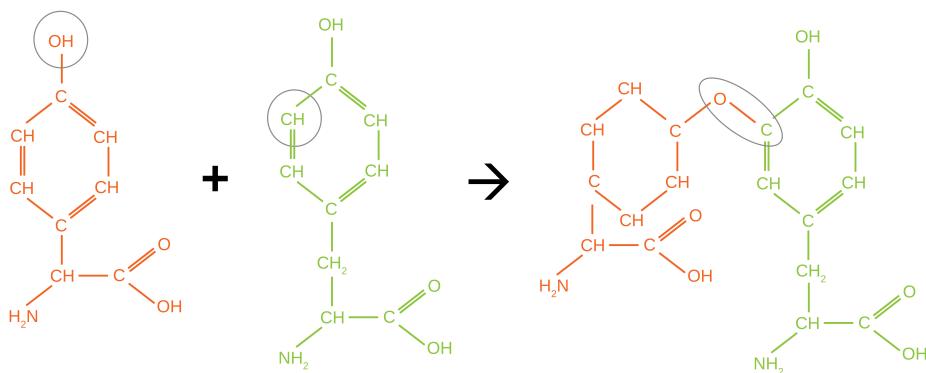


Figure 1.6: Exemple de liaison effectuée au sein d'un cycle benzène

de la catégorie des sucres (Glucose ou Arabinose par exemple). Les sucres sont inclus sous leur forme cyclisée dans les NRP. Suite à la liaison, un groupement OH a été perdu sur un carbone voisin de l'oxygène dans le cycle (voir figure 1.7). Face à eux, le monomère complémentaire ne perdra lui qu'un hydrogène pour former ainsi une molécule d'eau expulsée.

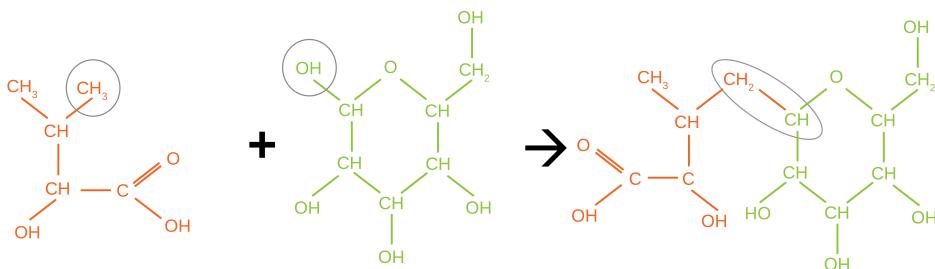


Figure 1.7: Exemple de liaison effectuée avec un sucre

Cette diversité dans la façon de lier les monomères accompagnée du nombre de sites de liaisons candidats explique la diversité structurelle présente au sein des NRP. La diversité du nombre de structures se voit aisément au sein des peptides représentés sur la figure 1.8.

1.1.2.4 Les activités

Les NRP présentent une grande diversité d'activités, souvent cruciales pour leurs producteurs. Beaucoup de NRP recensés sont des antibiotiques comme par exemple, la célèbre

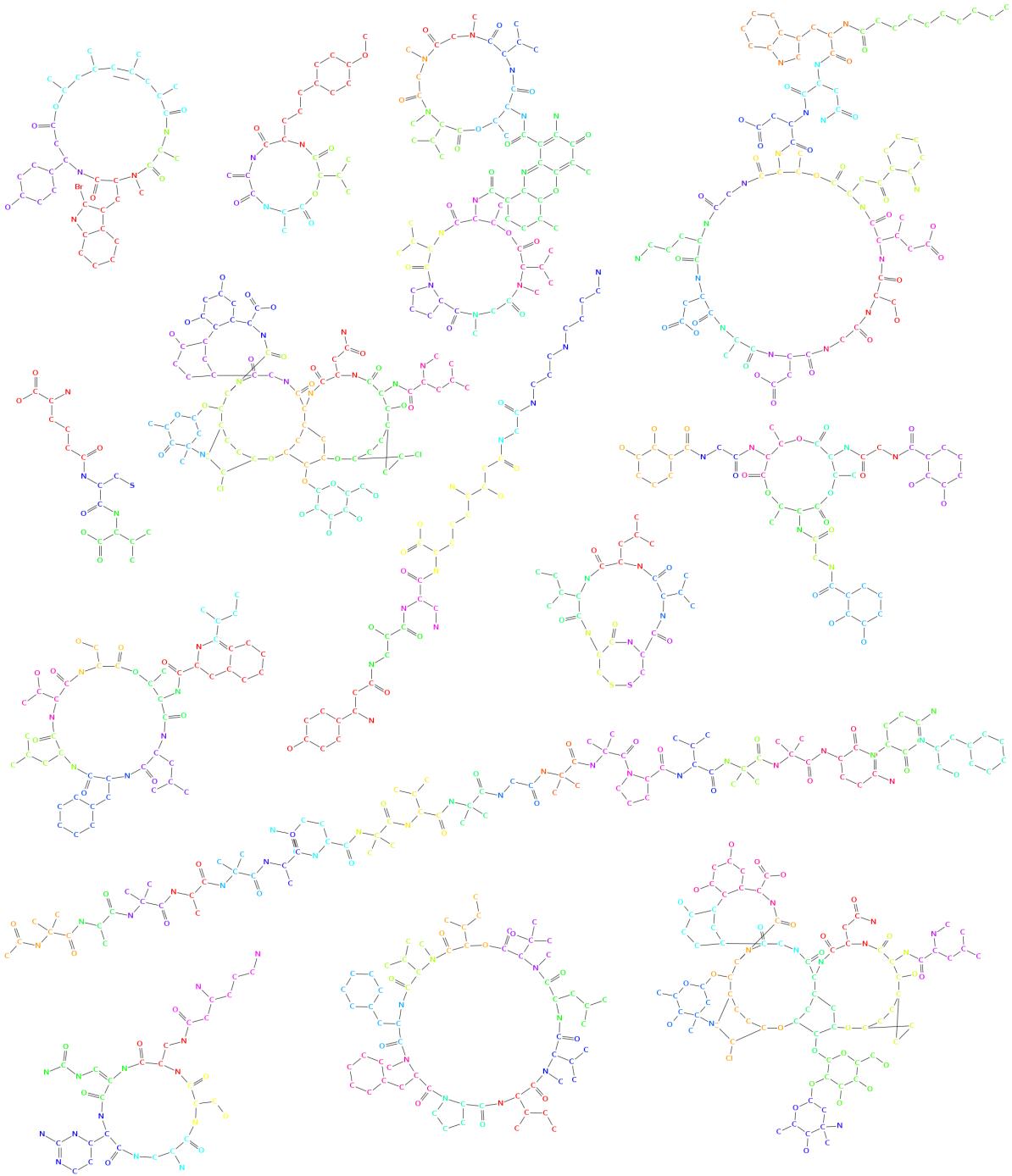


Figure 1.8: Patchwork de NRP

molécule de pénicilline synthétisée à partir d'un précurseur NRP appelé ACV [51]. La diversité d'activités peut être expliquée par les grandes variations de structures et de compositions par rapport aux peptides ribosomiques. Les structures particulières et les divers

monomères imposent aux NRP d'opter pour des repliements différents des protéines classiques. Les contraintes structurelles permettent des repliements spécifiques qui ne sont pas possibles au sein des peptides classiques. Les activités spécifiques aux NRP sont obtenues grâce à ces repliements et aux propriétés physico-chimiques des nombreux monomères.

Il est à noter qu'effectuer un assemblage non ribosomique est très consommateur en énergie pour l'organisme producteur. Les complexes enzymatiques à créer sont énormes et les gènes portant leur information peuvent représenter plusieurs pourcents de l'ADN chromosomal. De plus, contrairement au ribosome, une NRPS ne peut produire qu'un seul NRP (sauf quelques exceptions qui produisent des variants proches). Cependant les avantages obtenus grâce aux activités des NRP semblent avoir préservé ces gènes au cours du temps.

1.1.3 Généralités sur les synthétases

Comme nous l'avons vu dans la courte introduction précédente, les synthétases qui assemblent les NRP sont des protéines issues d'une synthèse classique. Chaque NRPS est un complexe enzymatique extrêmement grand, dont la taille est parfois comparable à celle d'un ribosome. Cependant, contrairement à un ribosome, les NRPS sont spécialisées. Une NRPS ne peut produire qu'un seul type de NRP. Elles sont organisées en modules consécutifs qui capturent, modifient et intègrent un monomère au NRP [37, 57]. Chacun de ces modules peut à son tour être subdivisé en domaines fonctionnels. Généralement un module est constitué de 3 domaines principaux et éventuellement de domaines dits optionnels [22]. Un module standard est d'abord composé d'un domaine de condensation (C) effectuant la liaison du peptide déjà formé au monomère en cours d'incorporation, puis d'un module d'adénylation (A) permettant la capture du monomère à insérer et d'un module de thiolation (T) effectuant la fixation du monomère capturé et le transfert du NRP depuis le domaine C précédent au domaine C suivant. Il se peut qu'un ou plusieurs domaines modifiant le monomère inclus, soient présents entre le domaine A et le domaine T. Le tout premier module d'une NRPS est différent du fait qu'il n'y a aucun monomère à lier auparavant. Il est l'une des exceptions à la règle de succession des modules C, A puis T.

Prenons pour exemple le NRP appelé HC-toxin [1]. Le HC-toxin est un peptide non ribosomique composé de 4 monomères. D'après la base de données MiBig [40] couplée à celle du NCBI [44], la synthétase permettant sa création est composée de 5218 acides aminés. Cette enzyme peut être séparée en 4 modules, incluant chacun un monomère. Chacun des modules est également subdivisé en domaines. Le découpage pour ce NRP est représenté sur la figure 1.9. Aux 4 modules correspondent 4 domaines A, 4 domaines T et 4 domaines C.

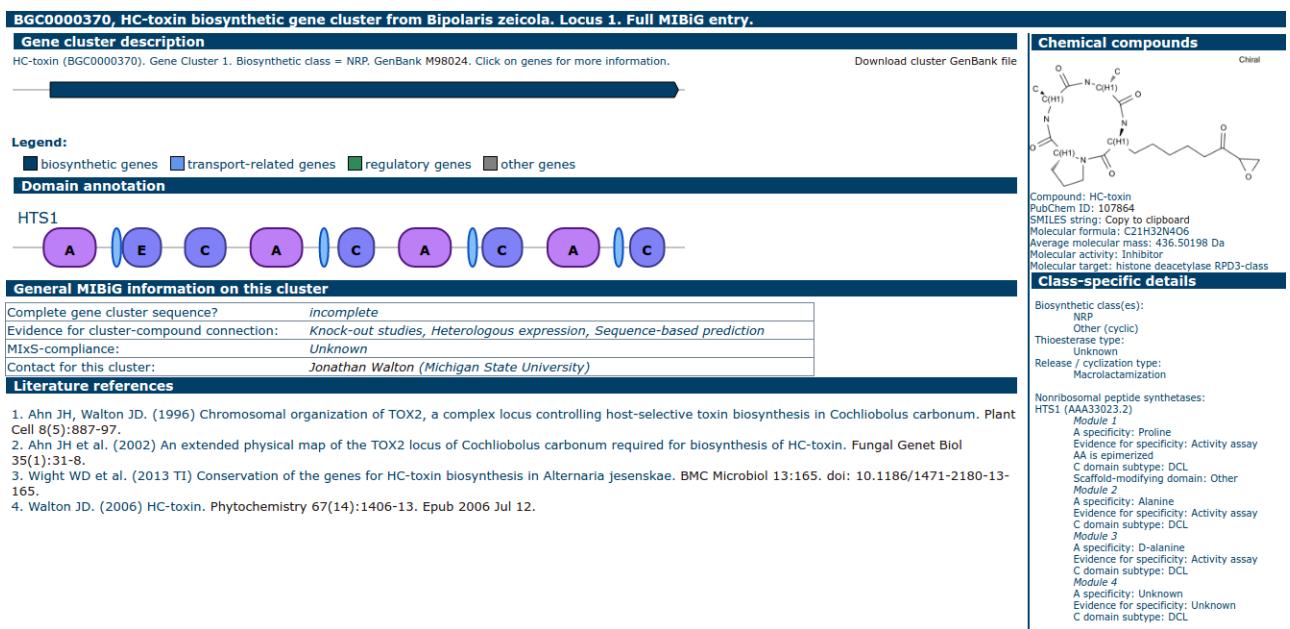


Figure 1.9: Modules NRPS pour la création du peptide HC-Toxin. Image issue du site web MiBIG. Les ovales bleus ciels représentent les domaines T.

On peut également voir un domaine d'épimérisation (modification de l'acide aminé inclus) présent en fin de premier domaine. Dans le cas de cette NRPS, le domaine C en fin d'enzyme effectue la cyclisation du peptide.

1.1.4 Les domaines principaux

1.1.4.1 Le domaine d'Adénylation (domaine A)

Le domaine d'adénylation (A) réalise la capture dans l'environnement d'un type de monomère donné. Il porte ainsi la spécificité du module dans lequel il se trouve. La séquence d'acides aminés de ce type de domaine varie fortement d'un domaine A à l'autre et seuls 16% de séquence du domaine sont conservées (sur environ 200 acides aminés codant un domaine) [65]. Bien que seuls quelques domaines A aient été caractérisés, on suppose qu'il en existe un grand nombre (potentiellement plusieurs centaines) puisque plusieurs centaines de monomères différents sont inclus par ces domaines.

Une fois le repliement de la synthétase effectué, les domaines A possèdent une cavité interne effectuant la reconnaissance et l'accueil d'une molécule. Les acides aminés de la synthétase formant cette cavité vont "électionner" le monomère à capturer dans l'environnement. La taille de la cavité ainsi que les propriétés physico-chimiques de ces acides

aminés en contact permettent une affinité forte avec un unique monomère. Il arrive parfois que le domaine ne soit pas spécifique à un seul monomère mais plutôt à un petit nombre de monomères très proches les uns des autres. Cependant, ce cas reste rare et, dans la majorité des cas, une NRPS ne crée qu'un seul NRP.

En 1999, Stachelhaus *et al.* publient un article analysant la structure de nombreux domaines A [65]. Ils alignent un grand nombre de domaines A pour en extraire les zones conservées et les zones spécifiques au monomère à capturer. Il apparaît tout d'abord que peu d'acides aminés sont conservés d'une structure à l'autre. Cependant ces acides aminés permettent de définir le “squelette” d'un domaine A. Ces alignements permettent également d'identifier une dizaine d'acides aminés qui paraissent spécifiques au monomère inclus. À partir de ces acides aminés spécifiques, les auteurs déterminent un codage pour chaque monomère qui deviendra le codage de référence. Ce codage est aujourd’hui appelé *code de Stachelhaus* et permet de mettre en relation une séquence d'un domaine A et le monomère spécifiquement recruté.

1.1.4.2 Le domaine de Thiolation (domaine T)

Le rôle du domaine T est d'assurer le transport du peptide en cours d'elongation au sein du module [12, 63]. Ce domaine est également souvent appelé “Peptidyl Carrier domain”. Chaque domaine T possède un groupement *4'-phosphopantetheinyl* incluant un “bras flexible” qui agrippe et déplace les monomères. Dans un premier temps, ce bras se plie vers le domaine A voisin pour se lier de manière covalente au monomère. Dans un second temps, le monomère est mené au site de condensation afin d'être inséré en bout du NRP. Enfin, le bras se plie dans la direction du domaine de condensation suivant afin de lier le NRP au monomère suivant. Durant cette étape, le peptide est libéré du domaine T courant pour être laissé à la charge du domaine T suivant.

1.1.4.3 Le domaine de Condensation (domaine C)

Le domaine C est par convention considéré comme le premier domaine de chaque module [64]. C'est un domaine qui lie le morceau de peptide assemblé par les modules précédents avec le monomère capturé par le domaine A du module courant. Bien que sa fonction principale reste omniprésente, il existe plusieurs variants [53] de ce domaine que nous allons décrire par la suite.

Les domaines $^L\text{C}_\text{L}$ et $^D\text{C}_\text{L}$ Les acides aminés capturés par les domaines A existent dans deux configurations différentes dites configurations L (Levogyre) et D (Dextrogyre).

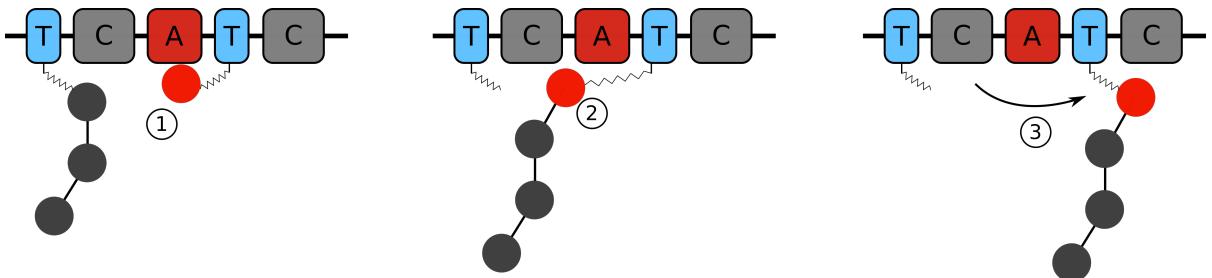


Figure 1.10: Rôle du domaine T au sein d'une NRPS. 1 - Liaison entre le monomère capturé par le domaine A et le bras du module T. 2 - Liaison entre le monomère et la chaîne peptidique en cours d'elongation. 3 - Déplacement du peptide vers le domaine C suivant

Ces configurations seront un peu plus détaillées en section 1.1.5.1. Il existe deux domaines C différents pour lier ces molécules. Le plus standard, lie deux monomères de type L. Le domaine est alors appelé domaine ^LC_L. Le second lui, lie un monomère D à un L, il est alors appelé domaine ^DC_L. Ces domaines sont juste deux spécialisations du modèle décrit ci-dessus.

Les domaines C starter (Cs) et C terminal (Ct) Comme leurs noms l'indiquent, ces domaines sont positionnés aux extrémités des synthétases. Ils ne sont donc pas entourés de deux modules différents mais sont inclus respectivement dans le premier ou dernier module. Il ne peuvent donc faire la liaison entre deux monomères fixés sur deux bras de domaines T.

Le domaine Cs capture et lie directement un monomère de l'environnement afin de débuter le peptide. Cela permet d'inclure par exemple les acides gras car ils ne sont pas reconnus par les domaines A. Ces modules capturent toujours le même type de monomères mais on ne sait actuellement pas exactement comment cette spécificité est encodée dans l'enzyme.

Le domaine Ct est un domaine de cyclisation uniquement présent chez les champignons. Dans certains peptides, ce domaine effectue une liaison entre le premier et le dernier monomère du peptide en cours de formation, créant ainsi un cycle[24].

Le domaine Cy Cette dernière variation du domaine C lie deux monomères voisins en effectuant deux liaisons qui forment un cycle local (voir figure 1.11). Cette liaison est toujours effectuée entre un monomère possédant un atome de soufre près d'un groupement amine (la Cystéine et certains de ses dérivés) et un monomère avec un groupement carboxyle.

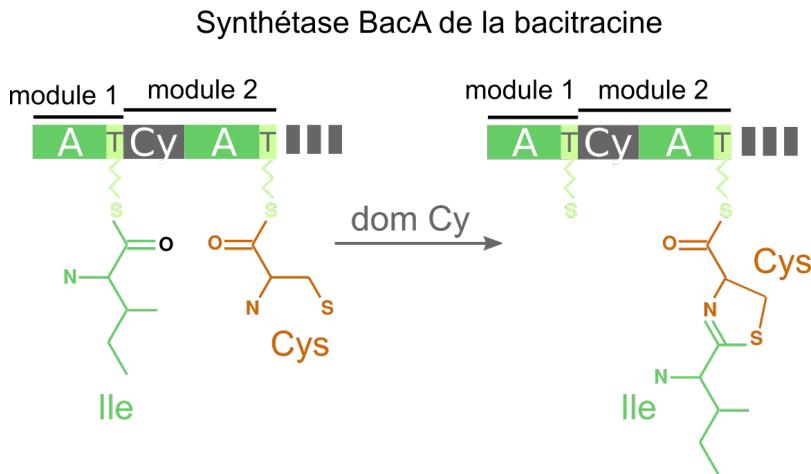


Figure 1.11: Cyclisation entre deux monomères par un domaine Cy

1.1.4.4 Le domaine de Thioestérase

Le domaine Te fait partie du module de terminaison. Il est le dernier domaine de ce module et vient succéder au domaine T ou aux domaines optionnels présents après le T. C'est un domaine de terminaison qui permet le décrochage et parfois la cyclisation du peptide complètement formé [33, 69]. Ce domaine de libération n'est pas toujours présent et sa fonction est parfois assumée par d'autres domaines comme le domaine Ct.

1.1.5 Les domaines de modification des monomères

Nous allons à présent décrire quelques domaines optionnels modifiant un monomère capturé par un domaine A au cours de la synthèse peptidique. Ces transformations permettent aux organismes d'obtenir des configurations monomériques peu fréquentes dans leur environnement.

1.1.5.1 La fonction d'épimérisation (domaines E et C/E)

Lors de la description des domaines ^LC_L et ^DC_L, nous avons rapidement évoqué la possibilité pour un monomère d'être dans deux conformations différentes tout en possédant pour autant la même composition et structure atomique. Ces deux conformations sont symétriques l'une de l'autre. Elles sont appelées Levogyre (L) et Dextrogyre (D).

Alors que la majorité de la vie est "gauchère" (ne contient que des acides aminés L), les NRPS autorisent l'inclusion de monomères sous forme D par la capture d'un monomère

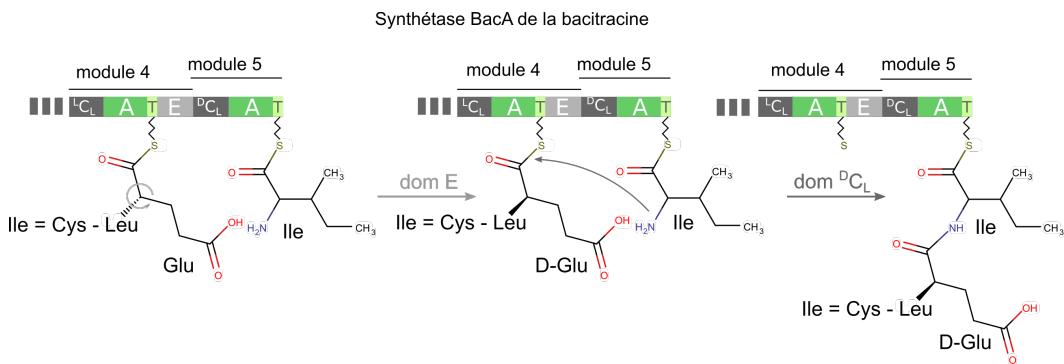


Figure 1.12: Épimérisation d'un monomère par un domaine E

L qui est ensuite transformé dans sa configuration D. Les domaines responsables de ces transformations sont appelés domaines d'*épimérisation* [12]. Une fois le monomère L lié au reste du peptide par le domaine C, il est emmené auprès du domaine E qui va changer son orientation. Le fait que cette transformation soit postérieure à la fixation sur le reste du peptide explique l'absence de domaines ^DC_D (Les monomères de droite seront toujours présentés au domaine C avant transformation). Il arrive parfois que cette transformation du monomère ne soit pas portée par un domaine E mais directement incluse dans le domaine C précédent. Ce domaine est alors appelé domaine C/E et assure les deux fonctionnalités [9, 83]. Ceci semble spécifique à quelques genres de bactéries.

1.1.5.2 Domaine de N-méthylation (NM)

De la même manière que le domaine E, le domaine NM agit à la suite de la liaison du monomère actuel à la chaîne peptidique précédemment assemblée. Ce domaine ajoute un méthyle (CH_3) sur le groupement amine du monomère courant.

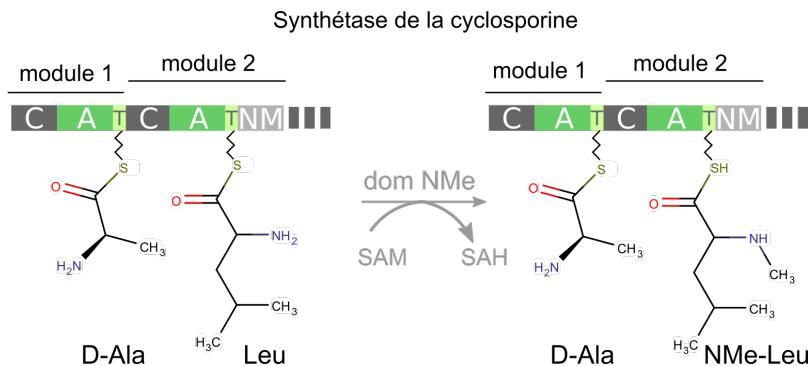


Figure 1.13: Méthylation d'un monomère par un domaine NM

1.1.5.3 Domaine de formylation

Ce domaine n'existe a priori que pour les modules d'initiation. Il permet la formylation (ajout d'un groupe C(=O)H) sur le groupement amine du premier monomère [58].

Synthétase de la gramicidine linéaire

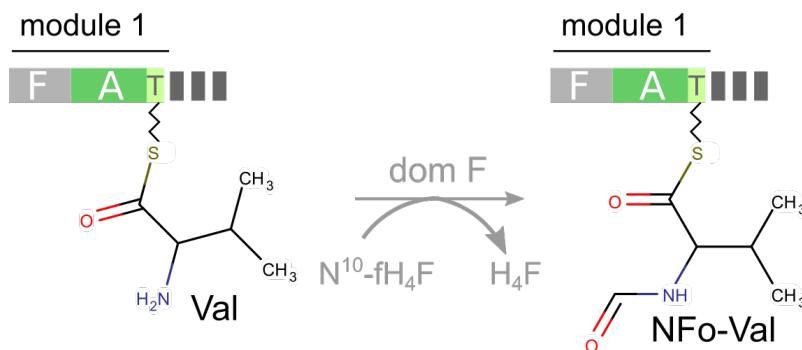


Figure 1.14: Formylation d'un monomère par un domaine F

1.1.6 Incorporations extra-NRPS

Depuis le début de cette partie, nous parlons de synthèse non ribosomique. Cependant, il arrive que le produit délivré par la synthétase et le produit final trouvé dans l'environnement de la cellule ne soient pas les mêmes. Lorsque cette différence est constatée, c'est que le peptide a subi une transformation entre la fin de sa synthèse non ribosomique et le moment où il est effectivement en activité. Toutes les modifications ponctuelles sont effectuées par des *enzymes de décoration* (voir section 1.1.6.1). Parfois, il arrive que ce soit une partie complète de la molécule qui ne soit pas NRP. Il existe également des molécules très proches des NRP appelés polyketides qui peuvent venir s'hybrider avec des NRP (voir section 1.1.6.2). Dans cette partie, nous allons aborder ces deux types de modification/ajout d'un NRP.

1.1.6.1 Les enzymes de décoration

Les clusters de gènes de NRPS ne sont pas uniquement constitués des gènes d'enzymes NRPS. De nombreux gènes accessoires sont également présents, généralement inclus dans le cluster. Parmi ceux-ci, certains gènes servent à modifier les NRP déjà relâchés par la NRPS. La *vancomycine* est un très bon exemple de peptide subissant des modifications enzymatiques post-synthèse.

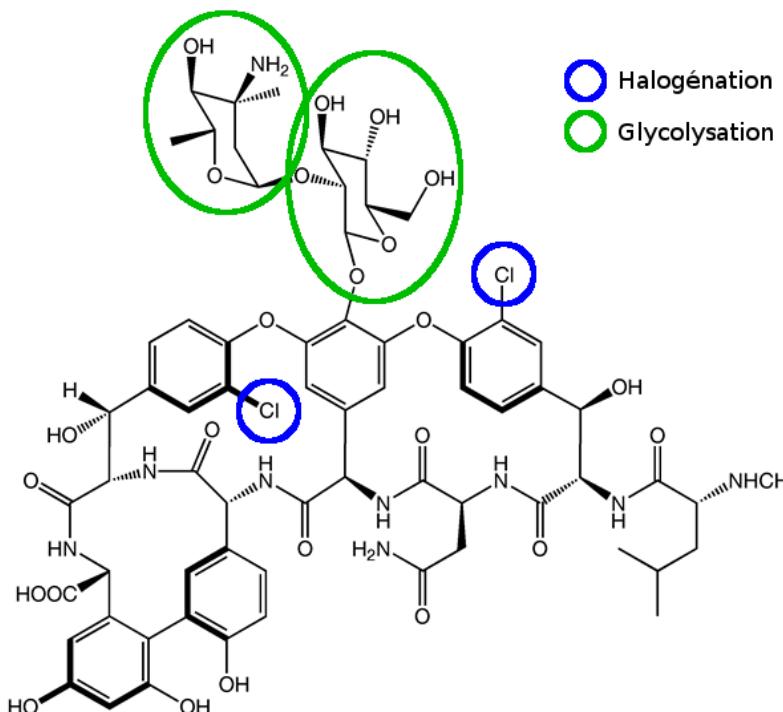


Figure 1.15: Modifications post-synthèse NRP pour la vancomycine

Sur la figure 1.15, les deux structures entourées en vert se démarquent par rapport au reste de la molécule. Ce sont deux sucres qui ont été ajoutés après la synthèse (une *vancosamine* et un *D-glucose*). Les sucres sont des monomères qui ne peuvent pas être capturés par des domaines A. Ce sont des gènes extérieurs à la NRPS mais faisant partie du cluster, qui sont dédiés à leur capture, transformation et liaison avec le NRP. Ce phénomène d'ajout de sucre est appelé *glycosylation*. Comme le suggère A. van Wageningen [72], trois enzymes entrent en action pour incorporer la vancosamine. La vancosamine n'étant pas naturellement présente dans l'environnement de l'organisme producteur, l'une des enzymes est dédiée à la création de ce sucre à partir du *NDP-4-keto-6-deoxyglucose*.

La vancomycine possède également deux atomes de chlore. Ces deux atomes sont inhabituels pour des NRP et ne sont pas initialement présents au sein de leurs monomères de base. Il n'existe pas non plus de domaine NRPS capable d'inclure ces atomes. A. van Wageningen dans l'article précédemment cité, montre également que deux gènes d'enzymes présents dans le cluster de la NRPS sont responsables de ces incorporations. Ce type d'enzyme est appelé enzyme d'*halogénéation* car il est capable d'inclure un atome halogène (fluor, chlore, brome, iodé)

Pour résumer, les enzymes de décoration post-synthèse sont capables d'effectuer 3

tâches que nous avons découvertes sur l'exemple de la vancomycine :

- La synthèse de monomères en modifiant des composants de l'environnement (par exemple, création de vancosamine)
- La création de liaisons entre monomères de l'environnement et NRP (par exemple, les sucres de la vancomycine)
- La modification de monomères déjà inclus dans le NRP (halogénéation par exemple)

Toutes ces enzymes augmentent à nouveau la combinatoire des molécules potentiellement créées.

1.1.6.2 Les PKS

Les PKS (Polyketide Synthase) [60, 66] sont, tout comme les NRPS, des enzymes modulaires synthétisant des polymères complexes par assemblage de monomères. Cependant, contrairement aux NRPS, les PKS n'incorporent que 4 monomères différents contenant chacun moins de 10 atomes. De la même manière que les domaines C, A et T des NRPS capturent des monomères spécifiques, les domaines ACP, At et KS condensent, capturent et lient les monomères. La diversité des PK vient, elle, du nombre de modifications effectuées par les nombreux modules optionnels. Tout comme les NRP, les PK sont des molécules très coûteuses en énergie pour la production, mais très utiles pour les organismes qui les synthétisent. Sur la figure 1.16 nous montrons le processus de synthèse d'une PK.

Il existe également des hybrides NRP-PK. Les clusters de gènes associés sont alors composés de gènes NRPS et de gènes PKS. Il existe alors deux moyens de créer ces hybrides. Soit la PK est créée séparément du NRP et les deux sont liés post-synthèse par des enzymes similaires aux enzymes décoratrices décrites précédemment. Soit la PKS et la NRPS sont produites simultanément et s'assemblent pour former un gros complexe enzymatique hybride. Le NRP-PK est alors directement formé à partir de la synthétase hybride en faisant passer le polymère en formation de module en module (qu'ils soient NRPS ou PKS). À nouveau, cette hybridation avec d'autres types de polymères augmente la quantité de molécules différentes possibles produites par des NRPS.

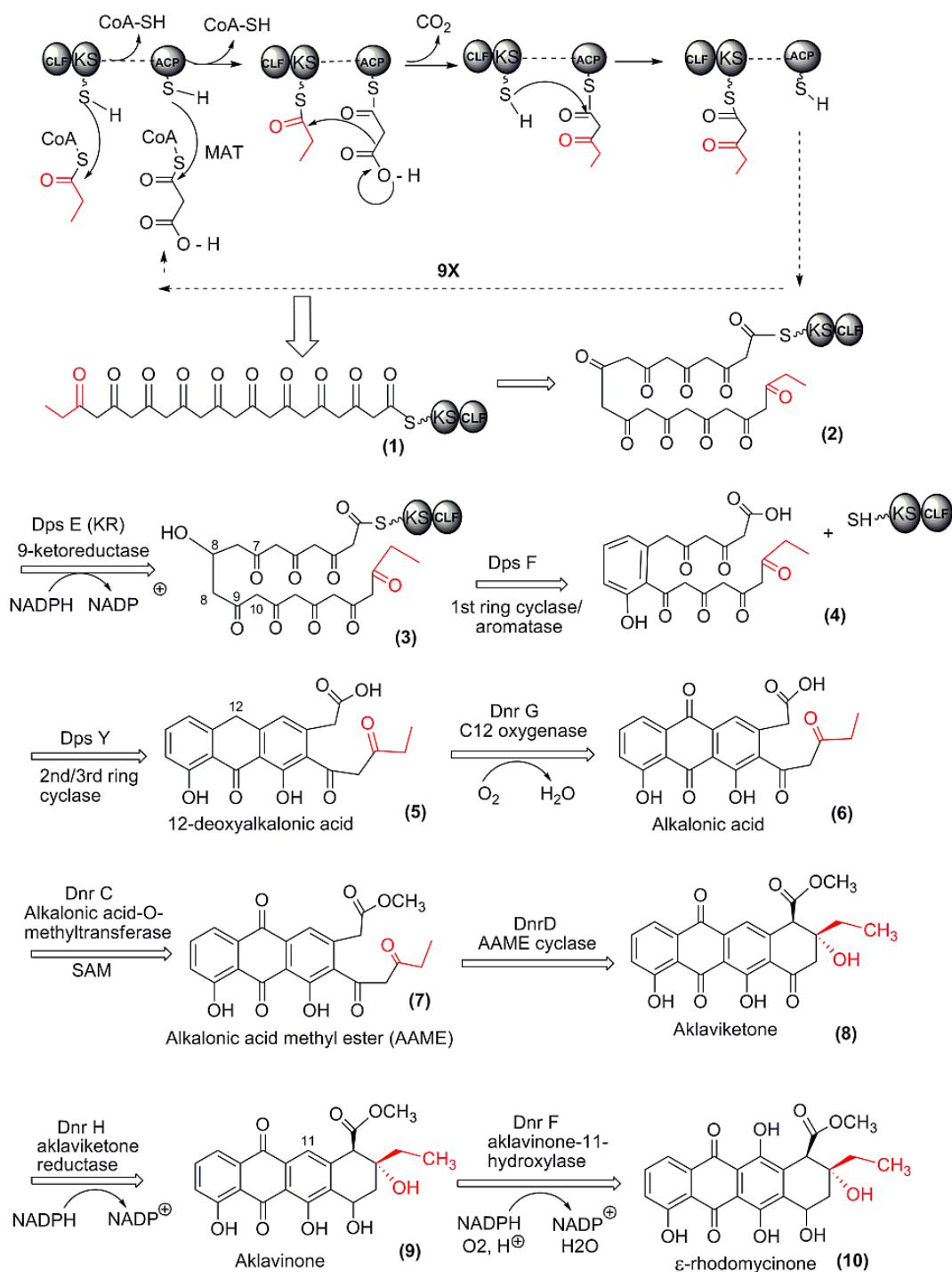


Figure 1.16: Synthèse d'une epsilon-rhodomycinone. Comme pour les NRPS, les PKS sont constitués de modules fonctionnels. Sur cet exemple, on peut voir que certains modules sont utilisés itérativement. La longue chaîne carbonée de la première étape est créée par une utilisation, répétée 9 fois, du même module. Tout comme les NRP, les PK peuvent être modifiés après libération par le PKS (étapes 5 à 10 ici). Source de l'image : en.wikipedia.org/wiki/Polyketide_synthase

1.2 Les outils bioinformatiques pour l'annotation et l'analyse des NRP/NRPS

Après avoir présenté les NRP ainsi que leurs voies de synthèse, nous allons nous attarder sur les outils bio-informatiques qui les ciblent. Les outils que nous allons présenter ici se focalisent, soit sur les NRPS, soit sur les NRP, c'est pourquoi la présentation séparera distinctement ces deux types d'outils. Pour chacune de ces catégories, nous allons également distinguer deux sous-catégories. La première portera sur les outils effectuant l'annotation des données. Pour les NRPS, l'annotation correspond à la phase de recherche de gènes NRPS dans le génome ainsi qu'à l'identification des différentes briques contenues dans ces gènes (entre autre les modules). Pour les NRP, l'annotation correspond à la phase d'identification et de décomposition des structures monomériques présentes dans les molécules. La seconde sous-catégorie présentera les moyens d'archivage et de mise à disposition des informations récoltées par les outils d'annotation. Cette partie rassemblera les différentes bases de données, dédiées ou non, et contenant une ou plusieurs informations à propos des NRPS ou NRP. Cette liste a été constituée à l'aide du portail des outils bioinformatiques dédiés aux métabolites secondaires [75].

1.2.1 Les outils d'annotation de NRPS

Dans cette partie, nous allons présenter les outils bioinformatiques permettant la découverte et l'annotation de NRPS à partir du génome. Cette recherche pourra être effectuée par étapes successives en commençant par la détection des clusters de gènes, en poursuivant par l'annotation des domaines des NRPS et en finissant par les prédictions spécifiques aux domaines précédemment détectés.

1.2.1.1 Détection de clusters de gènes

L'information des clusters de gènes est très importante pour comprendre les formes finales des NRP. La détection des clusters nous permet d'identifier l'ensemble des gènes impliqués dans la synthèse d'un NRP, c'est à dire l'ensemble des gènes NRPS ainsi que des gènes accompagnateurs tels que les gènes pour la création des enzymes de décoration. Avoir accès à l'ensemble des gènes d'un cluster facilite, par exemple, la détection des modifications post-synthèse qui peuvent modifier un NRP.

Il existe de nombreuses techniques de recherche de clusters de gènes. Certains outils génériques comme CLUSEAN [74] détectent tous types de clusters. Il existe également

des techniques spécialisées pour détecter directement des clusters de gènes de métabolites secondaires (les NRP font partie des métabolites secondaires). Par exemple, le logiciel CASSIS [79] exploite des propriétés de régions promotrices pour découvrir et classifier ces clusters.

De nombreux autres logiciels encore plus spécialisés existent. Il est aussi possible de découvrir des clusters de métabolites secondaires chez des bactéries [16], des champignons [32] ou encore plus précisément chez des champignons filamenteux [7, 71]. Ces outils sont très nombreux et dans la majorité des cas, une étude d'un organisme particulier passe par l'utilisation d'un logiciel spécifique à celui-ci.

1.2.1.2 Détection et identification de domaines NRPS

La détection et l'identification des gènes codant pour les domaines est au coeur de l'annotation des NRPS. Nous allons ici présenter des méthodes identifiant les différents types de domaines (A, C, T, ...) ainsi que des méthodes inférant les spécificités de ces domaines.

La recherche par alignement Il est possible de rechercher des domaines NRPS *quasi-manuellement* [8]. Pour cela, il faut tout d'abord constituer une petite base données de domaines NRPS très généralistes puis les rechercher dans le génome cible à l'aide d'algorithme d'alignement comme BLAST. En analysant ensuite manuellement les alignements obtenus, il est possible de détecter certains domaines qui auraient été éliminés par des algorithmes automatiques. Cette méthode reste très limitée au regard de la quantité d'informations à gérer. Elle n'est utilisée que dans des cas très particuliers où l'on suppose fortement la découverte de nouveaux éléments. Cette technique peut également être utilisée automatiquement. Après obtention des différents alignements, il est possible d'annoter le génome analysé par les domaines mappés les plus proches.

NRPS predictor NRPS predictor [54, 56] est un outil de prédiction des substrats capturés par les domaines A. Il analyse une séquence trouvée par un algorithme de détection de modules A et prédit sa spécificité. Cet outil est basé sur la recherche par Machines à Vecteurs de Support (Support Vector Machine -SVM-). L'idée est d'utiliser des données de domaines A déjà annotés afin de créer des séquences (vecteurs) caractéristiques des domaines. Ces vecteurs sont composés des propriétés physico-chimiques des acides aminés proches (à moins de 8Å) de la "poche" d'accueil du monomère. L'outil prédit les domaines A avec différents niveaux de granularité, ce qui donne de l'information même lorsque la spécificité précise du module A n'est pas détectée. Le niveau le plus fin correspond à la détection exacte du monomère mais il est également possible de prédire les spécificités par "groupements" de monomères. Par exemple, l'outil essayera de prédire si le monomère in-

clus est polaire, hydrophobe, aliphatique... Chacun de ces critères dispose également de son propre profil SVM. Cet outil est bien plus performant que le simple code de Stachelhaus et obtient une F-mesure de 80% pour la prédiction la plus fine et 95% pour les plus gros groupements de monomères.

NaPDoS et l'identification des domaines C NaPDoS [84] est un logiciel issu d'un travail sur la phylogénie des domaines C NRPS. Les auteurs ont créé une base de domaines C de NRPS déjà connus. À partir de cette base, ils ont aligné tous les domaines les uns contre les autres et construit une phylogénie. Comme Raush *et al.* en 2007 [53], cette phylogénie a permis aux auteurs de diviser les domaines C en plusieurs groupes et ainsi de créer une classification de ces domaines. Ces groupes ont été transformés en différentes HMM représentatives utilisées pour la classification des domaines au travers du serveur web de l'application.

2metDB - PKS/NRPS web server Ces deux logiciels sont deux implémentations de la même technique de détection des domaines NRPS et PKS [8]. La différence entre les deux est que 2metDB est un outil standalone alors que PKS/NRPS web server est une version en ligne. Les auteurs font le constat qu'un domaine NRPS ou PKS possède à la fois des zones très variables en composition, ainsi que des zones très conservées. Ces zones conservées déterminent des coordonnées fixes entre domaines A et permettent ainsi de localiser les atomes impliqués dans le code de Stachelhaus. Il existe des zones conservées au sein de tous les domaines. Les auteurs des logiciels ont tiré parti de ces zones en créant des HMM pour représenter des archéotypes pour chacun des types de domaine. Ces logiciels sont extrêmement rapides pour les recherches grâce à l'utilisation d'un faible nombre d'archéotypes.

antiSMASH, le pipeline pour l'identification de métabolites secondaires antiSMASH [39, 76] est sans doute le logiciel qui est venu, ces dernières années, révolutionner la prédiction de métabolites secondaires à partir de l'ADN. Le logiciel est un pipeline incluant de nombreux outils dont certains de ceux que nous avons décrits précédemment. Les outils sont lancés automatiquement les uns à la suite des autres en analysant les séquences avec une granularité de plus en plus fine. A un instant donné, en fonction des spécificités des génomes et des annotations réalisées précédemment sur le génome en cours d'analyse, antiSMASH va choisir les logiciels à exécuter pour affiner les prédictions. Par exemple, en lançant une analyse, antiSMASH va commencer par la recherche de clusters de gènes, puis identifier parmi les clusters détectés, ceux qui permettent la production de métabolites secondaires pour finir par lancer les outils spécifiques à chaque métabolite comme NRSPredictor et NaPDoS sur les clusters NRPS. Si le type de génome a été renseigné, le

1.2 LES OUTILS BIOINFORMATIQUES POUR L'ANNOTATION ET L'ANALYSE DES NRP/NRPS

27

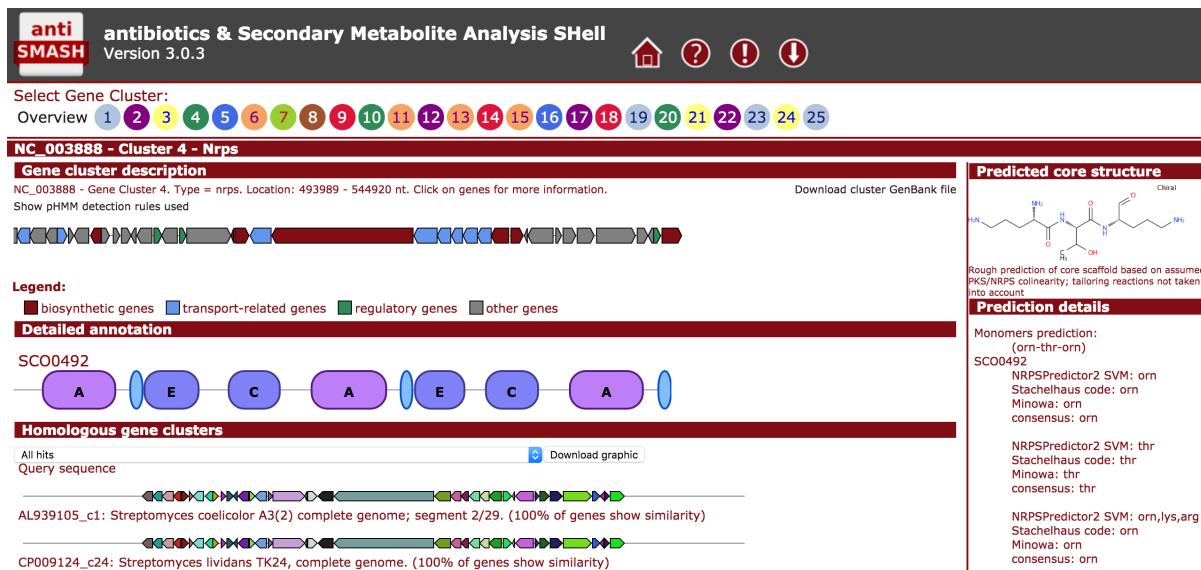


Figure 1.17: Exemple de résultat d’antiSMASH pour un cluster NRPS au sein du génome de Streptomyces coelicolor

pipeline lancera des outils spécifiques aux bactéries ou aux champignons.

L’un des avantages d’antiSMASH est sa facilité d’utilisation. Disponible en ligne, il permet l’envoi de génomes ou séquences protéiques pour une analyse sur des serveurs qui lui sont dédiés. Grâce à ces serveurs, l’utilisateur n’a pas de contraintes sur la machine nécessaire au calcul. Après parfois plusieurs heures de calcul, l’utilisateur reçoit un email contenant un lien vers ses résultats. La page de résultats comporte les résultats de tous les clusters qui ont été annotés avec des détails sur chacun des métabolites secondaires attendus. Cette facilité d’utilisation et la puissance des outils utilisés ont procuré un grand succès au pipeline auprès de la communauté. Le succès a été tel que les auteurs ont du recourir à des listes d’attente avec priorité pour pouvoir obtenir des résultats depuis le serveur dédié. Cependant, il faut bien rappeler que les structures issues du pipeline sont des prédictions et qu’elles sont donc à utiliser avec prudence.

1.2.2 Les bases de connaissances de NRPS

Nous allons ici décrire les bases de données spécifiques aux NRPS. Ces bases servent aussi bien au stockage des données ayant été générées par des logiciels de prédition, que des données ayant été découvertes par des analyses biologiques en laboratoire. Ces données sont en général produites indépendamment par des chercheurs s’intéressant à un sujet particulier, mais il est très important de les centraliser pour pouvoir en tirer des règles générales.

ClusterMine360 ClusterMine360 [14] est une base de données recensant des clusters de gènes NRPS et PKS. Cette base stocke uniquement les annotations NRPS (136 annotations de NRPS et hybrides NRPS-PKS) et pointe vers le NCBI pour toutes les références aux séquences. Les gènes de la base sont pour la plupart prédisés depuis antiSMASH. C'est d'ailleurs les résultats d'antiSMASH v1 qui sont affichés pour le détail de chaque cluster. Malheureusement cette base n'est pas tenue à jour et le dernier article posté en ligne date de 2013.

ClustScan Database ClustScan Database [18] est une base de données créée à partir des résultats de l'outil de prédiction de NRPS et PKS appelé ClustScan. Certains clusters ont été vérifiés à la main et sont accompagnés de publications lorsque les utilisateurs de la base les ont ajoutées (113 annotations de NRPS et hybrides NRPS-PKS). Le logiciel prédit également des structures NRP. Cependant, la prédiction s'arrête aux structures linéaires et à quelques formes cycliques. Ces données sont toutes des prédictions de structures NRP et ces structures devront donc être utilisées avec précaution. La base ne semble pas avoir été mise à jour récemment mais aucune date explicite n'apparaît sur le site web pour confirmer cette impression.

Minimum Information about a Biosynthetic Gene Cluster (MIBiG), la base de référence des clusters de gènes NRPS/PKS MIBiG [40] est une base de données de clusters de gènes de métabolites secondaires, incluant des gènes NRPS accompagnés de leurs annotations (376 annotations de NRPS et hybrides NRPS-PKS). Les créateurs de cette base cherchent avant tout la qualité de l'annotation plutôt que la quantité de données. En effet, le seul moyen d'ajouter des données est de les entrer manuellement en indiquant précisément leur provenance. Sur le site de MIBiG, les annotations de NRPS sont souvent accompagnées des peptides qui leur correspondent. Ce lien fort entre clusters de gènes et produit est très intéressant mais n'est disponible à ce jour que pour 67 entrées.

1.2.3 L'annotation de NRP

Annotations automatiques depuis l'ADN Le séquençage de polymères autres que l'ADN/ARN est une tâche difficile. C'est pourquoi les structures des protéines ne sont pas directement analysées mais inférées depuis les séquences d'ADN. Dans le cas des NRP, l'interprétation de l'ADN indique uniquement les séquences protéiques des NRPS. Il est nécessaire ensuite de comprendre précisément le fonctionnement de ces NRPS pour prédire le NRP depuis leur séquence. Cependant, comme nous l'avons vu lors des descriptions de logiciels d'analyse de NRPS, cette compréhension ne donne pas encore de résultats parfaits et complets. Les logiciels ne prédisent pas précisément les monomères capturés par

les domaines A. Par exemple, il est possible d'obtenir une prédition de type “c'est un monomère polaire” sans avoir plus de détails. Il est également possible d'obtenir des structures monomériques incomplètes, en manquant par exemple les composés capturés par des domaines C starters. Enfin, en l'état actuel de nos connaissances, il est impossible de prédire la forme exacte pour des structures non linéaires.

L'annotation automatique de NRP depuis l'ADN est une méthode très rapide grâce à la quantité de données issues des technologies récentes de séquençage mais dont les résultats peuvent être incertains voire incomplets. Lorsque cette technique est utilisée pour inférer la structure d'un NRP, il sera toujours nécessaire de la vérifier expérimentalement.

Analyse par spectrométrie de masse La spectrométrie de masse est une technique physique d'identification de structures moléculaires. Ce procédé est basé sur la mesure de masses moléculaires. On utilise un échantillon purifié d'une molécule d'intérêt puis on envoie un niveau d'énergie précis afin de casser la molécule en morceaux. Selon les technologies, une ou plusieurs cassures accompagnées de une ou plusieurs mesures sont effectuées. En sortie, un spectre composé de différents pics représentant les masses des fragments est obtenu. Ces masses peuvent ensuite être comparées à des bases de données de masses caractéristiques pour ainsi connaître les composants. Enfin, à partir des morceaux identifiés, il est parfois possible de reconstruire la molécule originelle (mais cela dépend fortement des technologies utilisées).

Cette méthode permet d'obtenir une reconstruction atomique fiable lorsque toutes les masses sont présentes dans les bases de référence. Ces bases donnent par exemple les masses des acides aminés classiques et de leur dérivés proches. Si la technologie de spectrométrie utilisée casse les molécules au niveau de leurs liaisons peptidiques, la reconstruction peut plus facilement aboutir à une annotation monomérique de toutes les parties composées d'acides aminés classiques. Cependant, les NRP étant composés de plus de 500 monomères différents, l'annotation monomérique directement issue des analyses de spectres n'est pas suffisante. Beaucoup de monomères, n'étant pas proches de molécules classiques, ne sont pas répertoriés parmi les masses classiques. De plus, les peptides comportant des liaisons non peptidiques ne peuvent pas être facilement découpés par certaines technologies de spectrométrie. Pour finir l'annotation d'un NRP, il est nécessaire d'avoir recours à un spécialiste du domaine afin qu'il nomme chacun des monomères non annotés.

L'outil d'analyse CycloBranch [46] essaie, lui, de remonter directement à la structure monomérique des NRP. En essayant de casser les molécules uniquement au niveau des liaisons peptidiques attendues, le logiciel génère des spectres qu'il analyse ensuite informatiquement. Ayant constitué une base de données de masses de monomères NRP, il arrive

à contourner une partie du problème de manque de masses de référence. Les auteurs construisent ensuite une multitude de graphes monomériques possibles en essayant différentes formes branchées et cyclisées. Puis, pour choisir parmi les différents peptides probables, ils utilisent une méthode de génération de spectres théoriques afin de les comparer au spectre réel. Cette génération de spectres attendus élimine de nombreux candidats et mène parfois vers une structure monomérique unique. Cependant, la reconstruction unique ne représente pas la majorité des cas. De plus, les résultats obtenus dans l'article ne semblent pas tous être reproductibles (communication personnelle).

Pour résumer, la spectrométrie de masse donne accès à des structures atomiques de molécules avec un degré de certitude très élevé et parfois à des reconstructions de structures biologiques NRP fiables [41]. De plus, contrairement à l'annotation depuis l'ADN, nous sommes certains que la molécule est produite puisqu'elle est directement extraite de l'environnement étudié. Cependant, le besoin d'une expertise spécifique pour terminer les annotations et le faible débit de génération de données dû aux purifications nécessaires, ne permettent pas une annotation efficace.

Analyse en tandem ADN-Spectrométrie

Parmi les logiciels d'analyse de spectres de masse pour la découverte de NRP, quelques uns se démarquent en croisant les informations aux prédictions génomiques. C'est notamment le cas des logiciels NRPQuest [42] et "Genomes-to-Natural Products platform" (GNP) [?]. Dans les deux cas, les logiciels construisent une annotation NRP depuis l'ADN, via un ou plusieurs logiciels de prédiction externe, puis essayent de relier ces prédictions aux spectres de masse. La différence entre les deux logiciels est au niveau du lien qui est fait entre les deux sources d'informations. NRPQuest construit des spectres candidats depuis les prédictions ADN pour les comparer aux spectres de masses réels. De son côté, GNP essaie de mettre en correspondance des pics du spectre avec des prédictions de molécules en utilisant divers algorithmes d'apprentissage. En mettant ainsi en relation les deux informations, ils arrivent à reconstituer des compositions et structures NRP plus fiables que chaque technique séparée. Cependant dans les deux cas, les reconstructions NRP sont très fortement dépendantes des algorithmes de prédiction depuis l'ADN. Même avec des spectres de très bonne qualité, rien ne sera détectable après de mauvaises prédictions NRPS. C'est une limitation forte de ces logiciels que les auteurs de NRPQuest pointent eux mêmes à la fin de leur article.

Analyse par Résonance Magnétique Nucléaire La Résonance Magnétique Nucléaire (RMN) est une technique de détermination de structures moléculaires. Cette technique exploite la propriété qu'ont certains atomes à émettre un rayonnement lorsqu'ils

sont placés dans un champ magnétique. Les structures obtenues en sortie sont uniquement atomiques. Selon la technologie de la machine utilisée, la structure 3D de la molécule peut être accessible. Il est donc également nécessaire d'avoir recours à un expert pour effectuer une annotation monomérique. Cette technique est également très coûteuse de par les équipements nécessaires pour réaliser les analyses ainsi que le temps nécessaire pour créer un cristal capable de convenir à l'analyse. Enfin les quantités et la pureté nécessaires dans les échantillons ne sont souvent pas atteignables par des productions NRP.

1.2.4 Les bases de connaissance de NRP

Les bases de NRP sont très rares en comparaison des bases de NRPS. Quasiment toutes les bases de NRPS que nous avons précédemment présentées comportent également les structures de NRP prédicts. Cependant, comme nous l'avons déjà dit plusieurs fois, ces données ne sont pas toutes fiables. Beaucoup des structures sont inexactes, voir incomplètes, car elles sont issues de prédictions.

Norine, la base de référence des NRP Norine [11, 23] est la base de données de référence pour les peptides non ribosomiques annotés. Cette base de données a été développée et est entretenue par l'équipe Bonsai de l'université de Lille 1, équipe au sein de laquelle j'effectue ma thèse, en collaboration avec l'équipe ProBioGem de l'institut Charles Violette. Contrairement aux autres bases, c'est une base entièrement dédiée aux NRP. Tous sont extraits de publications présentant les molécules et déterminant leur caractère NRP. Chaque molécule entrée dans la base est accompagnée de sa structure monomérique (et parfois de sa structure atomique). Depuis 2015, la base de données est entièrement ouverte aux contributions extérieures. En quelques clics et en appuyant sa soumission par des articles de preuve de la voie de synthèse, toute personne peut soumettre de nouvelles entrées. La base contient 1184 annotations de NRP. Les structures présentes dans cette base sont fiables, ce qui est un avantage comparé aux prédictions présentées jusqu'à présent. Cependant, le processus d'ajout d'informations, strict pour préserver la qualité, ne permet pas d'ajouter de nouvelles entrées par centaines. Soulignons tout de même le travail effectué par les auteurs initiaux qui ont entré plus de 1100 annotations à la main.

MiBiG Revenons sur la base MiBiG, base que nous avions présentée parmi les bases NRPS. MiBiG est la seule base, hors Norine, encore active et contenant des annotations vérifiées de NRP. Comme nous l'avions déjà évoqué, MiBiG est une base surtout dédiée à l'annotation de NRPS. Cependant, les annotations de NRPS sont souvent accompagnées d'informations sur les peptides produits. Ces informations peuvent être de deux types. Dans un premier cas, la structure du NRP produit est connue (par exemple via des

analyses de spectres). Dans ce cas, l'annotation NRPS a pu être validée par l'expérience et nous pouvons lui faire confiance. Dans le second cas, le NRP provient uniquement d'une reconstruction du NRP à partir des prédictions NRPS. Il se peut alors que le NRP ne soit que partiel et il est probable qu'il soit incorrect.

Chapter 2

Smiles2Monomers : Des atomes vers les monomères

2.1 Introduction

Pour comprendre un système biologique, il est nécessaire d'étudier les interactions chimiques qui s'y déroulent. Au delà du simple catalogage qui peut en être fait, comprendre les actions moléculaires permet d'utiliser ou même d'inventer de nouvelles substances pour l'industrie. Afin d'explorer ces activités, un postulat simple a été posé : deux molécules proches ont deux activités proches. Ce postulat simple paraît raisonnable puisqu'il s'appuie sur le fait que l'activité d'une molécule est portée par la configuration spatiale des différents éléments qui la composent. On sait aujourd'hui que ce postulat est en partie faux car on connaît quelques contre-exemples [49]. Cependant, dans la majorité des cas, cette approximation fonctionne bien et tout un domaine de chemo-informatique s'est développé dans ce sens. Toutes les techniques cherchant à rapprocher les structures des activités moléculaires sont regroupées derrière l'acronyme QSAR (Quantitative structure-activity relationship). Les techniques QSAR sont nombreuses et de types variés [3, 34, 49]. Certaines s'attardent sur les ressemblances de structures atomiques 2D ou 3D, certaines sur les propriétés magnétiques, d'autres sur les contraintes de repliement *etc.* Beaucoup ne se cantonnent pas à une seule des ressemblances citées ci-dessus mais essaient de les combiner pour prédire au mieux l'activité. Cependant, ces techniques ont deux limites. Premièrement, pour déterminer l'activité d'une molécule, il est nécessaire de connaître les activités de molécules "proches", qu'il faut donc avoir annotées auparavant. Deuxièmement, les temps de calcul nécessaires augmentent rapidement avec le nombre de critères à comparer et la complexité d'analyse de chacun de ces critères. Le but de ces méthodes est donc d'inclure d'abord les

critères les plus sélectifs afin de déterminer des activités connues en un temps raisonnable.

Beaucoup de NRP possèdent des activités très intéressantes (antibiotiques, anti-tumeurs, antidiouleurs, ...). À ce titre, nous cherchons en permanence à caractériser les activités de chaque nouvelle molécule découverte. Lorsque ces activités ne sont pas déterminées expérimentalement, les logiciels exploitant des techniques de type QSAR permettent d'effectuer une prédition. En 2012 puis 2014, Abdo et al. publient de nouvelles méthodes de prédition d'activité NRP, se basant sur les spécificités des NRP [4, 5]. Dans les deux cas, les prédictions sont effectuées à partir des compositions monomériques et non pas atomiques. Dans le premier article [4], la prédition est effectuée en créant pour chaque peptide connu un *fingerprint* et en ajoutant un fingerprint pour le peptide dont on cherche l'activité. Pour prédire l'activité, les auteurs utilisent plusieurs classificateurs présent dans la librairie WEKA [?]. Dans le second article [5], c'est un classifieur par réseau Bayésien qui est entraîné sur les compositions en monomères. Une fois les réseaux entraînés, il suffit de fournir une composition en entrée pour avoir les probabilités de chaque activité en sortie. Dans les deux cas, il est important de constater que l'abstraction monomérique est suffisante pour une prédition de qualité. De plus, les structures des molécules ne sont pas utilisées, ce qui laisse encore beaucoup de possibilités d'améliorations.

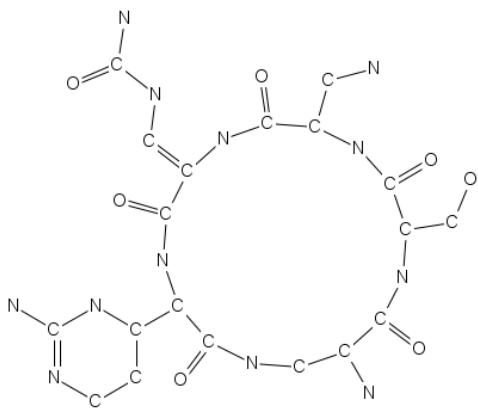
Ce que nous venons de montrer par cet exemple de prédition d'activité, c'est l'importance de la détermination des structures plus proches de la réalité biologique comme les structures monomériques. Connaître ce type de structure est un enjeu fort pour la détermination de caractéristiques moléculaires haut niveau (activité ou conformation 3D par exemple). Comme nous l'avons fait remarqué précédemment, il n'existe que peu d'outils permettant des annotations fiables de NRP. Les caractérisations fiables de NRP sont en très grande majorité effectuées manuellement par des biologistes et chimistes. Beaucoup de structures atomiques sont également connues sans annotations monomériques malgré leur présence dans de grandes bases de données. Il nous a donc paru nécessaire de créer un outil qui infère la structure monomérique à partir d'une structure atomique et c'est cet outil d'annotation que nous allons décrire durant ce chapitre.

2.2 Formalisation du problème d'annotation

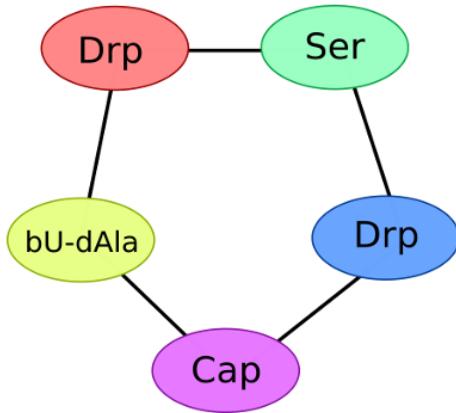
2.2.1 Définition du problème

Nous cherchons à créer un outil bioinformatique permettant l'obtention de nouvelles annotations monomériques NRP fiables. La génération d'annotations peut se baser sur la

recherche d'informations à partir des gènes ou à partir des molécules dont on connaît déjà la structure. Dans notre cas, nous n'avons pas choisi de démarrer des clusters de gènes NRPS puisque des modèles déjà bien éprouvés existent (NRPSpredictor, NaPDoS, ...). Les avancées dans ce domaines viendront avec le nombre d'annotations fiables de binômes clusters/NRP qui permettront l'amélioration des modèles d'apprentissage actuels.



Structure atomique



Structure monomérique

Figure 2.1: Structures atomique et monomérique de la Caprieomycin IIA.

Dans notre cas, nous allons partir des structures atomiques pour déduire les composants monomériques (voir figure 2.1). De telles structures sont présentes dans de nombreuses bases de données publiques, telles que PDB [?] ou PubChem [?]. Ces bases contiennent de nombreuses descriptions de molécules NRP sans annotations monomériques directement accessibles. Notre objectif sera d'être capable, pour toute structure moléculaire NRP, de déduire sa composition en monomères. Ainsi, toutes les structures atomiques présentes dans les bases généralistes pourront être annotées et ajoutées à des bases spécialisées telles que Norine. Ce type d'annotation pourra également être effectué pour mettre en relation les structures atomiques non annotées des bases généralistes avec les annotations biologiques présentes dans des bases spécialisées.

2.2.2 L'existant

Parmi l'existant, la méthode CHUCKLES [61] et le logiciel molBLOCKS [26] se démarquent des autres. Dans les deux cas, les logiciels permettent, à partir d'une structure atomique d'un polymère, de trouver la structure monomérique. Les deux logiciels sont opposés dans leur façon de traiter le problème. CHUCKLES est constructif car il part des monomères pour reconstituer le polymère alors que molBLOCKS est destructif car il part du polymère

pour retrouver les monomères par découpes successives de liens. Nous allons ici présenter ces deux approches.

2.2.2.1 CHUCKLES, une méthode constructive

CHUCKLES [61] est une méthode publiée en 1994 pour la recherche des peptides au sein d'une base de données. Le but est d'interroger une base de données de molécules en utilisant des expressions régulières entrées par l'utilisateur et représentant des sous-molécules d'intérêt. Les recherches peuvent être effectuées à la fois au niveau atomique et au niveau monomérique. Le but de CHUCKLES est de convertir rapidement les données du format monomérique vers le format atomique et inversement. Les auteurs pensent qu'il est plus utile de conserver les formats atomiques en base plutôt que les séquences protéiques. Sachant que la plupart des peptides de l'époque étaient déjà représentés par leur séquence, les auteurs proposent une méthode de transformation de la séquence monomérique vers la structure atomique. Cette transformation est appelée "Forward-Translation" (FT).

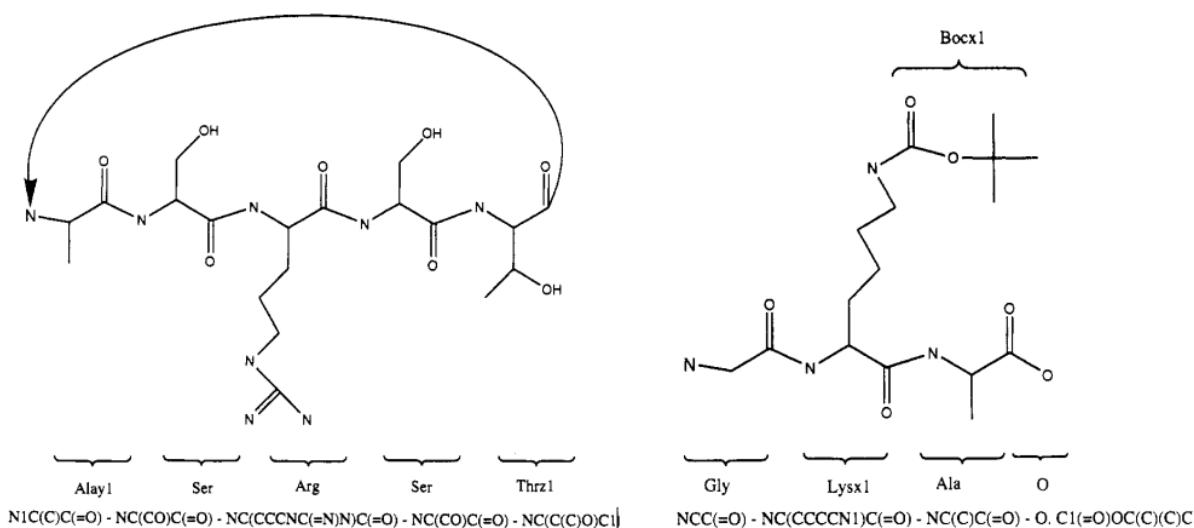


Figure 2.2: CHUCKLES Forward Translation. Transformation des annotations peptidiques en SMILES. Sous l'annotation peptidique est écrite la formule SMILES après concaténation de tous les SMILES des monomères trouvés. Schémas issus de la publication

Pour mener à bien chaque FT, les auteurs utilisent une table de transformation qui recense les structures atomiques de chacun des monomères à transformer. Chaque structure ne doit pas contenir les atomes qui vont être perdus lors des liaisons avec d'autres monomères. Pour les structures linéaires, la méthode propose une simple concaténation

des structures atomiques selon la chaîne peptidique. Lorsque les structures sont branchées ou cycliques, il est nécessaire d'ajouter dans la structure initiale l'information de liaison annexée.

Cette méthode de reconstruction des structures atomiques est très intéressante mais souffre du manque d'information sur le sens des monomères inclus. On peut remarquer que la reconstruction ne peut fonctionner à tous les coups sans cette information. En effet, si les monomères ne possèdent pas un groupement carboxyle et un amine, le sens naturel des liaisons peptidiques ne peut être retrouvé. La reconstruction se fait alors en utilisant un sens aléatoire pour le monomère.

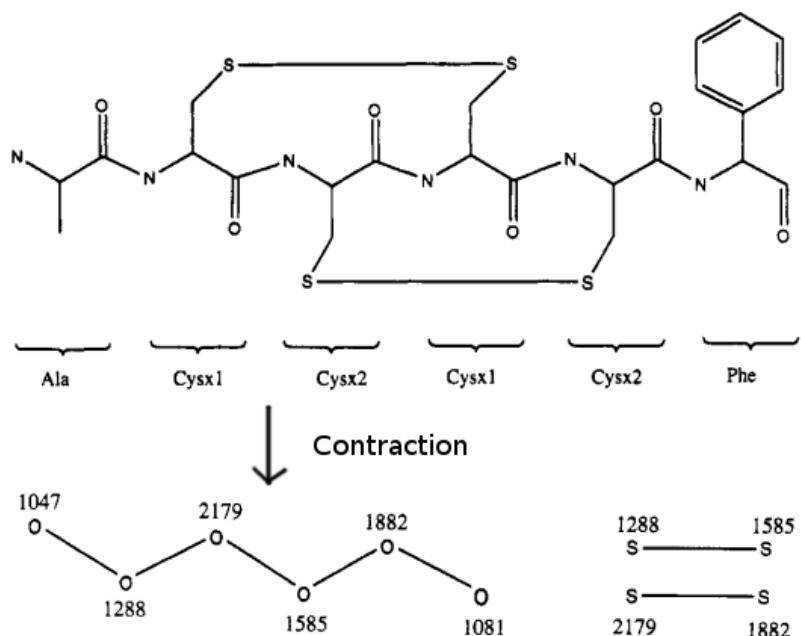


Figure 2.3: CHUCKLES Backward Translation. La première partie représente la phase de recherche des monomères au sein de la structure atomique. La seconde partie représente la structure monomérique post contraction. On peut voir à gauche la chaîne principale qui a été extraite à partir du N initial. À droite sont répertoriés les deux ponts disulfure extraits hors de la chaîne principale. Chaque monomère découvert comporte un identifiant différent de son nom pour éviter d'avoir deux fois le même nom comme cela pourrait être le cas ici (exemple avec Cysx1 et Cysx2). Schémas issus de la publication [61]

La seconde partie de la méthode, précisément celle qui nous concerne, présente la transformation inverse (“Backward-Transformation” -BT-). C'est cette partie qui est très intéressante pour l'annotation car elle part des structures atomiques pour reconstruire les structures monomériques. Les auteurs proposent une annotation en deux temps. Une phase de recherche des monomères, puis une conversion des monomères trouvés en une annota-

tion. L'idée de la première phase est de rechercher chaque monomère connu en masquant les atomes déjà utilisés par des monomères précédemment trouvés. En recherchant les monomères par taille (nombre d'atomes) décroissante, les auteurs s'assurent que de petits monomères ne viendront pas prendre la place de plus gros. C'est un algorithme glouton sans aucun retour arrière. Lors de la seconde phase, les atomes des molécules sont contractés pour ne plus former qu'un seul noeud par monomère. Le N terminal de la chaîne peptidique est ensuite recherché afin d'extraire l'annotation de cette chaîne en suivant les liaisons peptidiques. Enfin, les liaisons n'ayant pas encore été explorées sont ajoutées à la représentation monomérique finale.

Comme pour le FT, le BT utilise des techniques intelligentes mais encore faillibles. Dans le cas des NRP, certains monomères ne possèdent pas les deux domaines pour les liaisons peptidiques. Il est donc potentiellement impossible de retrouver une chaîne principale. De plus, lors de la recherche, si les monomères à rechercher sont structurellement proches, il est possible qu'un gros monomère vienne prendre la place d'un plus petit en débordant sur un ou plusieurs monomères voisins. Dans ce cas, vu qu'aucun retour en arrière n'est effectué, l'annotation sera inexacte voir incomplète.

2.2.2.2 molBLOCKS, une méthode destructive

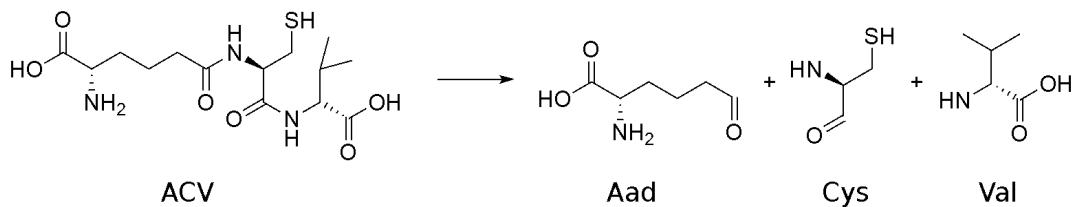


Figure 2.4: Application de la méthode molBLOCKs sur le polymère ACV. Le polymère est séparé en morceau par une règle de liaison peptidique (voir équation 2.1). Les trois morceaux correspondent aux trois monomères réellement inclus, la découpe fonctionne.

Le logiciel molBLOCKS [26] n'est, à la base, pas conçu pour le travail que nous cherchons à effectuer. C'est un logiciel qui permet de découvrir des monomères en repérant les fragments fréquents au sein d'une population de polymères. Pour cela, le logiciel découpe les polymères en petits fragments et effectue des comptages de ceux-ci. Ces découpes suivent des règles entrées par l'utilisateur. Par exemple en donnant comme règle

$$O = CNH \longrightarrow C = O + NH \quad (2.1)$$

toutes les liaisons peptidiques seront ciblées et déconnectées (voir figure 2.4). Pour des protéines ribosomiques, la découpe sera donc parfaite avec cette seule règle et les monomères seront découverts.

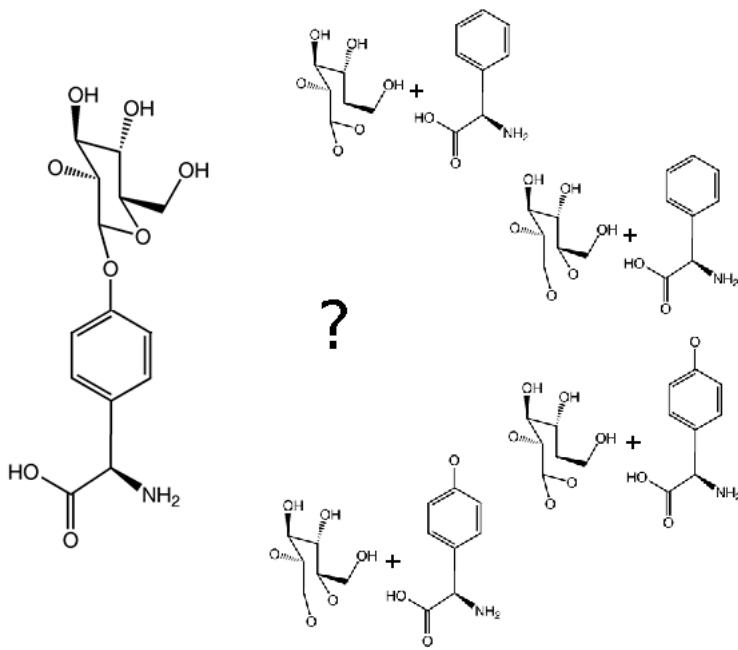


Figure 2.5: Dans le cas de ce dimère, un sucre est lié à une Hydroxy-Phenil-Glycine (Hpg). En utilisant la règle de découpe à appliquer pour les liaisons hydroxyle, une ambiguïté et une erreur apparaissent. Nous ne pouvons savoir de quel côté les découpes vont être les plus pertinentes avant d'essayer de reconnaître les monomères. De ce fait, 4 découpes différentes sont possibles. De plus, le cycle du sucre se voit ouvert alors que la découpe est inutile.

Dans notre cas, nous pourrions utiliser un tel logiciel pour découper les NRP en monomères. En effectuant par la suite une étape de reconnaissance de ces monomères, il serait alors possible de reconstruire le peptide. Cependant, certaines réactions effectuées au sein des NRP ne permettent pas de découper entièrement le peptide avec la seule règle énoncée précédemment. En effet, toutes les liaisons non peptidiques énoncées au chapitre précédent ne seront pas découpées. Il est donc nécessaire de rajouter de nouvelles règles. Cependant, la liaison hydroxyle à elle seule ne permet déjà plus d'effectuer une découpe correcte car pour pouvoir la délier, il faudrait une règle du type

$$COC \longrightarrow CO + C \quad (2.2)$$

Cette règle est cependant ambiguë. Il est impossible de connaître de quel côté du *O* la

liaison doit être rompue. De plus, certains monomères comme le glucose sont initialement composés de *COC*. Ces monomères seront donc découpés alors qu'ils ne devraient pas l'être. Pour ces deux raisons, nous ne pourrons pas directement utiliser ce logiciel afin de convertir nos structures atomiques en structures monomériques.

2.2.3 Vers des problèmes informatiques

Pour concevoir notre logiciel d'annotation, nous allons nous inspirer des techniques existantes en essayant de dépasser leurs limitations. Nous possédons, au sein de Norine, une grande base de connaissances de monomères existants (plus de 500) et nous pouvons nous appuyer sur cette base bien établie pour assurer l'existence des monomères lors des annotations. A partir de ces données, essayons de concevoir un algorithme constructif.

2.2.3.1 Une histoire de sous problèmes

La méthode CHUCKLES tire partie de la différence entre les 20 acides aminés principaux pour se permettre de rechercher les plus gros en premier et de ne jamais revenir sur les choix faits. Rechercher les monomères par taille (en nombre d'atomes) décroissante et en masquant les atomes déjà utilisés est un choix pertinent car il permet d'éviter de sur-détecter partout les petits acides aminés comme la glycine. Dans notre cas, la proximité entre certains monomères ne nous permettra pas de les départager grâce à un tri par taille. Nous risquons d'introduire de trop gros monomères prenant la place d'autres et créant artificiellement des "trous" dans l'annotation. Il sera nécessaire de modifier les solutions incomplètes pour essayer de les améliorer. Contrairement à CHUCKLES, nous aurons donc besoin de connaître les différents monomères candidats à la composition du polymère même lorsqu'ils sont chevauchants à des monomères déjà placés. Si d'un côté nous recherchons les monomères au sein du NRP, puis de l'autre nous assemblons ceux trouvés, alors nous pouvons séparer notre programme en deux sous-problèmes complètement distincts.

Dans un premier temps, nous allons chercher à déterminer les monomères qui pourraient être les candidats pour composer le peptide étudié. En utilisant la base Norine, nous pouvons indépendamment rechercher chacun des monomères connus au sein de la molécule à annoter. La recherche exacte d'un monomère au sein d'un peptide est donc notre premier sous problème à résoudre. Ce sous problème sera appelé la "recherche" (voir figure 2.6).

Dans un second temps, nous devrons choisir parmi tous les monomères candidats issus de la recherche, lesquels composent réellement le polymère. Nous aurons plusieurs

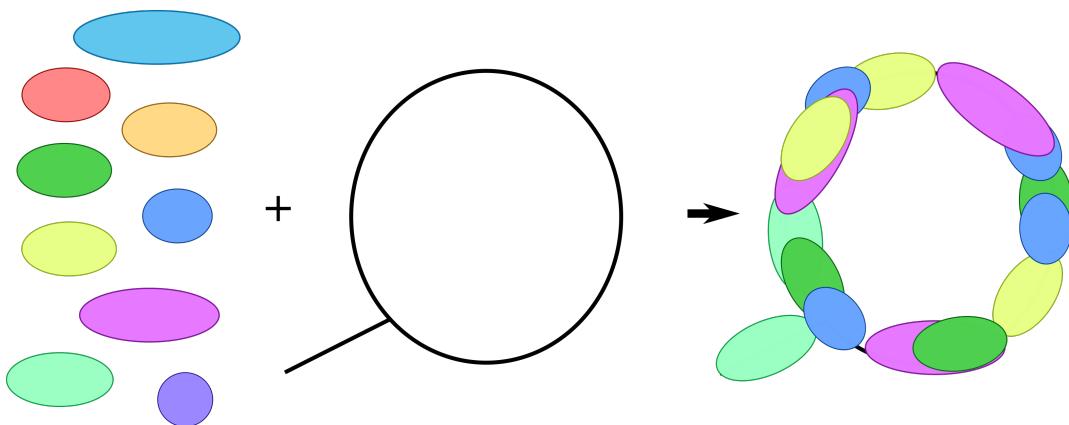


Figure 2.6: Première étape : Recherche des monomères (ellipses colorées) sur le peptide à annoter (cycle noir branché).

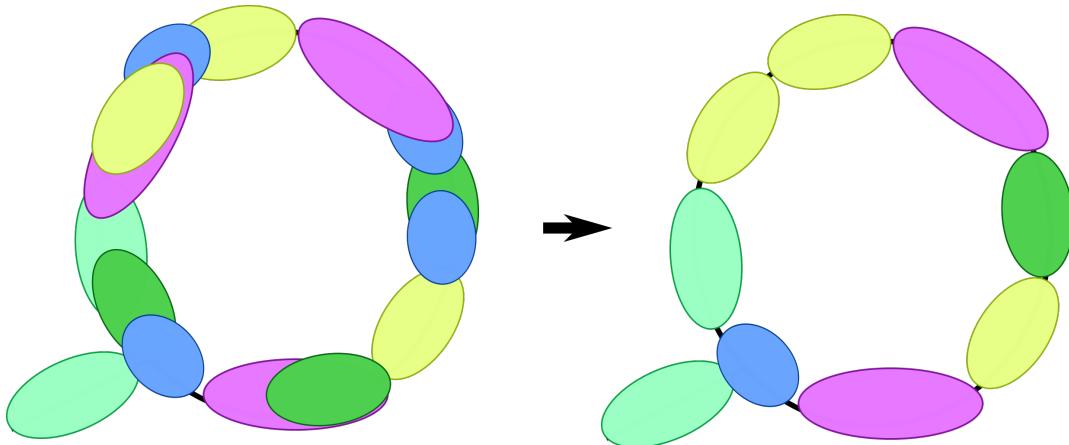


Figure 2.7: Seconde étape : Sélection des monomères parmi ceux trouvés à l'étape 1. Le but est d'obtenir une structure monomérique non chevauchante de couverture maximale.

contraintes associées à ce problème. Aucun des monomères choisis dans la solution ne peut être chevauchant avec un autre. En effet, aucun atome ne peut provenir de deux molécules différentes et donc aucune sous partie de monomères ne peut être chevauchante avec un autre monomère. De plus nous allons chercher à obtenir une annotation complète des peptides, ce qui correspond à maximiser les atomes du polymère couverts par des monomères. On appellera cette étape l'étape de “pavage” (voir figure 2.7).

2.2.3.2 Des molécules aux graphes

Pour comprendre la suite du propos, il est nécessaire de comprendre ce qu'est un graphe au sens informatique du terme. Pour plus de détails référez-vous au prérequis [0.3](#).

2.2.3.3 Formalisation en problèmes de graphes

Comme expliqué plus haut, le problème de recherche de sous composants d'un polymère peut être divisé en deux problèmes plus petits : un problème de recherche de monomères et un problème de pavage des monomères trouvés. Traitons donc séparément ces deux problèmes.

Recherche de monomères Rechercher un monomère dans un polymère revient à rechercher un petit graphe atomique S dans un plus gros graphe G . Deux cas sont alors possibles : soit nous cherchons à trouver exactement S dans G , soit nous cherchons une “approximation” de S dans G . Dans le premier cas, nous allons nous intéresser à l’“Iso-morphisme de Sous-graphe” (en anglais *Subgraph Isomorphism* -SI-). Dans le second cas, nous nous intéresserons au “Sous-graphe Maximum Commun” (en anglais *Maximum Common Subgraph* -MCS-). Les recherches de SI ou de MCS ont toutes deux été prouvées NP-Complet mais, pour chacun des problèmes, il existe des algorithmes rapides en pratique pour des graphes aux propriétés particulières. Nous aborderons cela en section [2.3](#).

Pavage des monomères Une fois les monomères détectés au sein du polymère, il est nécessaire d'effectuer un choix entre les différents monomères candidats se chevauchant sur un ou plusieurs atomes. L'objectif est donc de ne garder qu'un des différents monomères possibles tout en couvrant au mieux le polymère, et si possible dans sa totalité. Ce genre de méthode est appelée “pavage sans recouvrement”. Cette étape donnant accès à un lien direct entre atomes et monomères, une étape de contraction de graphe permettra, au final, d'obtenir le graphe monomérique. Nous aborderons le pavage en section [2.4.3](#).

2.3 Sous-graphe Maximum Commun vs Isomorphisme de Sous-graphe

Nous venons de voir que la recherche de monomères pouvait correspondre à deux problèmes différents en recherche de graphe. Formalisons et étudions ici ces deux problèmes afin de choisir celui qui nous utiliserons au sein de notre logiciel d'annotations. Voyons les différentes façons de les résoudre ainsi que les forces et faiblesses de ces approches.

2.3.1 Sous-graphe Maximum Commun

2.3.1.1 Définition

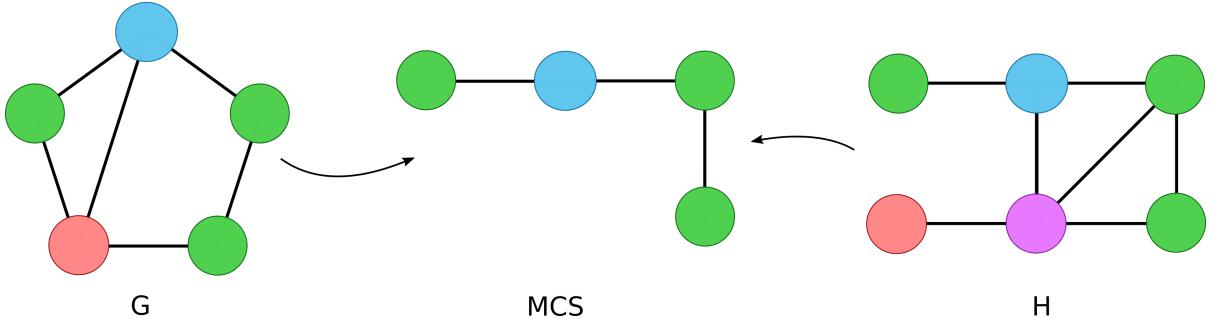


Figure 2.8: MCS obtenu à partir des graphes G et H. Le MCS obtenu ne contient pas le nœud étiqueté “rouge” du fait des liens vers ce nœud qui ne sont pas les mêmes dans G et H.

Commençons par définir précisément le problème du MCS. Comme le nom l’indique, le problème du MCS est un problème de recherche de la sous partie commune maximale entre deux graphes (voir figure 2.8). C'est-à-dire trouver un ensemble de nœuds présents dans les deux graphes et dont les arêtes qui les inter-connectent sont présentes dans les deux graphes. Il faut non seulement que les nœuds partagent les même étiquettes, mais également les mêmes voisins. Si deux nœuds ont la même étiquette mais ne partagent pas les mêmes voisins dans les deux graphes, alors ils ne font pas partie du MCS. C'est par exemple le cas du nœud rouge au sein de la figure 2.8. Le problème de recherche de sous-graphe commun connexe maximal (MCCS) est un problème encore plus difficile, car il impose de fortes contraintes supplémentaires. Le problème du MCS a été prouvé NP-Complet[25], ce qui laisse peu de perspectives pour l'émergence d'algorithmes de résolution polynomiaux exacts.

Dans notre cas, nous pouvons effectuer une réduction vers ce problème pour la recherche des monomères. Si l'on considère qu'un monomère et un polymère sont des graphes atomiques, alors trouver une sous partie commune à ces deux graphes revient à trouver un emplacement où une partie du monomère est retrouvée dans le peptide. En imposant que le graphe commun soit d'une taille minimale fixée, nous pourrons éviter d'avoir des résultats ne contenant que quelques atomes par hasard.

Du fait de la NP-complétude du problème, trouver des solutions exactes est très compliqué. Il existe tout-de-même quelques algorithmes de résolution exacts mais la majorité des articles à propos de MCS traitent d'heuristiques de résolution rapide. Voyons ensemble

ces deux catégories d’algorithmes en s’appuyant sur les articles de synthèses [20, 55].

2.3.1.2 Les méthodes exactes de résolution

Au sein des algorithmes de résolution exacte du MCS, il existe deux grandes écoles. D’un côté, le problème du MCS est réduit en un problème de clique maximale, avant d’utiliser des solveurs dédiés à ce second problème. D’un autre côté, les méthodes de résolution se basent sur des méthodes de recherche avec retour arrière (*backtracking*).

La réduction en clique maximale Il est possible de réduire le problème MCS en un problème de résolution de clique maximale [27, 50, 52]. Une clique est un ensemble de nœuds dans un graphe tel que tous les nœuds de cet ensemble sont reliés entre eux. Rechercher la clique maximale revient à rechercher un sous ensemble maximal de nœuds qui forment une clique. Il est possible d’obtenir plusieurs cliques maximales dans un même graphe. Ce problème est également NP-Complet [6]. Les résolutions de ce problème sont toutes des dérivées de la recherche exhaustive par backtracking. Il est possible de ne pas passer par toutes les combinaisons possibles en réduisant l’espace de recherche en fonction de l’arité des nœuds ou de leur contexte local mais malgré tout, les algorithmes restent exponentiels [68].

Pour obtenir un MCS à partir de solveurs de clique maximale, il est nécessaire de construire un graphe de compatibilité (voir schéma 2.9). Un graphe de compatibilité (GC) est un graphe créé à partir du croisement des deux graphes (qu’on appellera G et H). Pour les nœuds du GC, on génère tous les couples de nœuds (x, y) possibles avec x issu de G et y de H et dont les étiquettes sont identiques. Ensuite on ajoute une arête entre deux couples (x_1, y_1) et (x_2, y_2) de GC si x_1 est voisin de x_2 dans G et y_1 voisin de y_2 dans H. On ajoute également une arête si les x sont non voisins et les y également. En résolvant le problème de clique maximale sur le GC, on obtient le MCS.

Recherche par Backtracking La majorité des algorithmes exacts effectuent des recherches via des méthodes de backtracking [31, 36, 38]. Un MCS est une bijection entre deux sous-ensembles de nœuds dans deux graphes distincts. Les méthodes de backtracking effectuent une reconstruction itérative de cette bijection. Les couples de nœuds compatibles, n’étant pas encore présents au sein de la bijection, sont ajoutés un à un (voir figure 2.10). Lorsque l’algorithme arrive dans une impasse (plus aucune possibilité d’ajouter un couple), il revient en arrière sur une association puis fait un choix différent. En sauvegardant l’ensemble qui a contenu le plus de nœuds, on obtient la clique maximale après le parcours de tout l’arbre des possibilités. Encore une fois, certaines des techniques citées permettent des accélérations pratiques en coupant des branches de recherche mais jamais

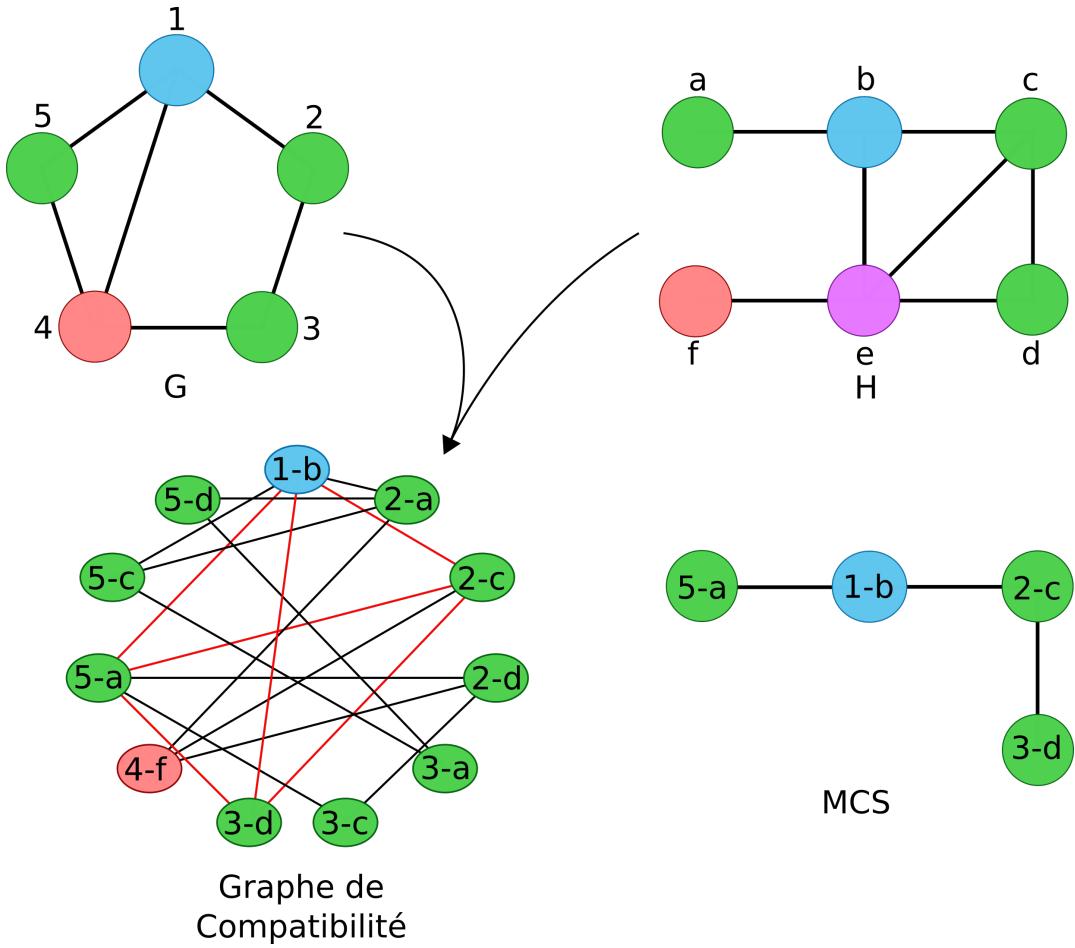


Figure 2.9: Résolution d'un MCS par réduction en clique maximale. Les couleurs représentent les étiquettes des nœuds. Dans le graphe de compatibilité les arêtes rouges identifient la clique maximale détectée.

l'ordre de grandeur de la complexité n'est changé. Dans le cas des MCCS, il faut inclure une contrainte supplémentaire [13]. Chacun des nœuds du couple à ajouter doit être voisin de l'un des nœuds d'un même couple précédemment inclus dans la bijection.

Il existe bien d'autres méthodes exactes pour résoudre des MCS pour des graphes particuliers (par exemple pour des quasi-arbres de *treewidth* borné [81]). Cependant, si nous souhaitons traiter n'importe quelles molécules, même avec les structures les plus extrêmes, nous ne pouvons utiliser ces méthodes. Nous ne détaillerons donc pas celles-ci ici.

2.3.1.3 Les méthodes heuristiques

Il existe une très grande quantité d'heuristiques de résolution du MCS. La méthode la plus utilisée est une méthode de recherche par similarités locales [78, 82]. Ces heuristiques

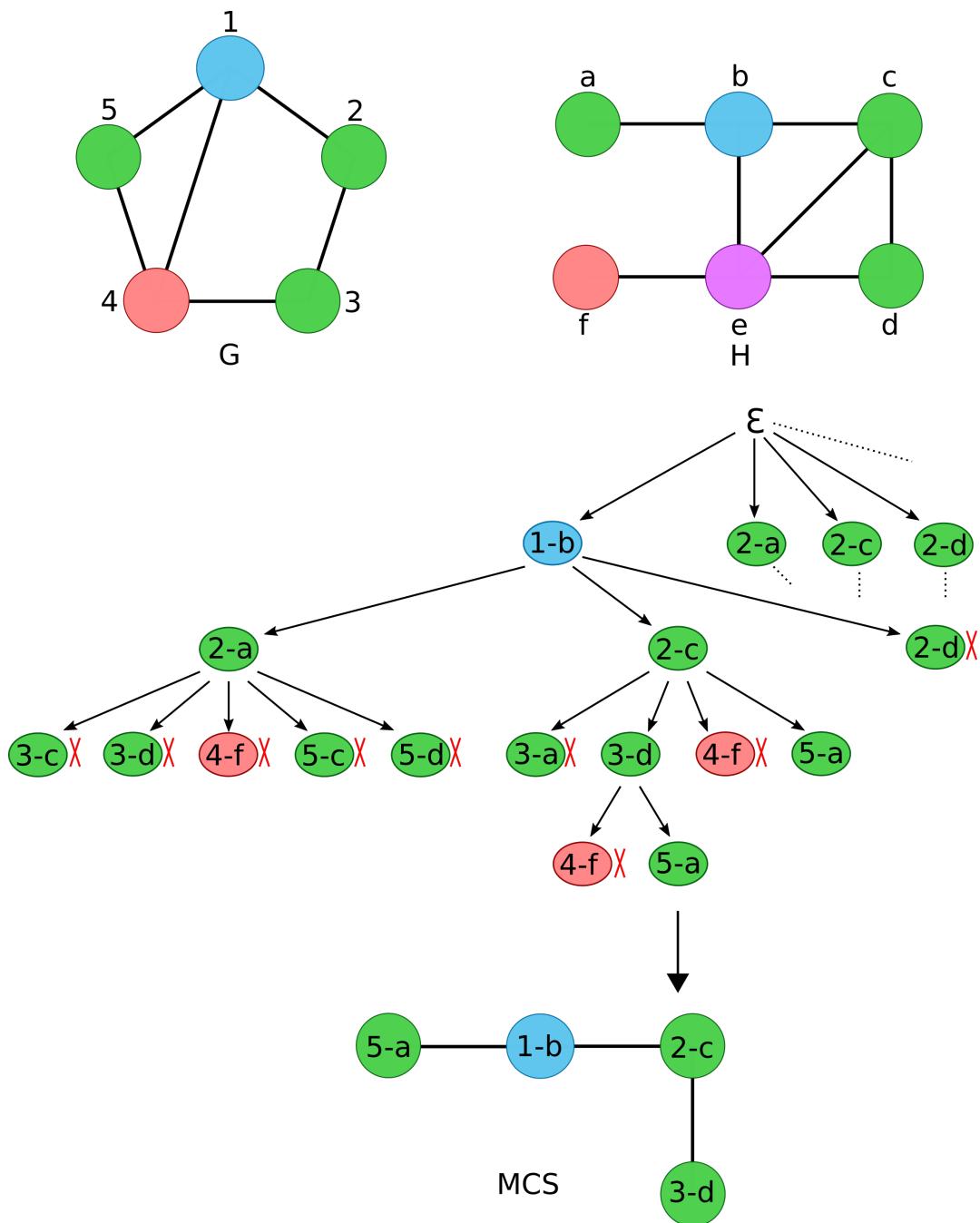


Figure 2.10: Exemple de résolution de MCS par un backtracking naïf. L’arbre part d’une bijection vide et ajoute un couple à chaque étape. Parfois l’algorithme tombe sur un couple compatible mais dont les voisins ne sont pas présents dans la bijection. La solution est alors rejetée (croix rouge). La solution est trouvée par l’ensemble des chemins qui mènent aux feuilles valides les plus profondes dans l’arbre.

sont utilisées au sein de bases de données pour rechercher des structures chimiques par ressemblance avec une requête. Chaque molécule de la base est décomposée en un grand nombre de critères topologiques locaux (l'arité des nœuds par exemple). Ces critères sont entrés dans un vecteur appelé *fingerprint*. Lors d'une requête, le motif recherché est à son tour transformé en *fingerprint* et une mesure de similarité est effectuée avec les vecteurs présents en base. Les *fingerprints* les “plus proches” (d'après une fonction de score [35, 45]) sont déclarés contenir une sous partie commune. Cette méthode ne donne aucune assurance quant à l'obtention de réels MCS. C'est la rapidité d'exécution de l'algorithme qui est recherchée. Dans notre cas, cette méthode est inappropriée puisque nous cherchons des MCS exacts.

D'autres méthodes se rapprochent plus des méthodes citées dans le paragraphe sur le backtracking [27, 73]. Ici, les auteurs construisent plusieurs bijections en parallèle. La différence avec les méthodes exactes est qu'ils ne reviennent jamais sur les décisions prises pour construire les MCS. Un nombre fixe de bijections est maintenu en permanence. Ces bijections ont été sélectionnées par une fonction de score comme étant celles qui avaient le plus de chance de mener à un MCS. Au final, les bijections trouvées ne seront pas forcément les vrais MCS. Cette seconde méthode est très intéressante lorsque l'on cherche un bon rapport entre rapidité et qualité du résultat.

2.3.2 Isomorphisme de sous-graphe

2.3.2.1 Définition

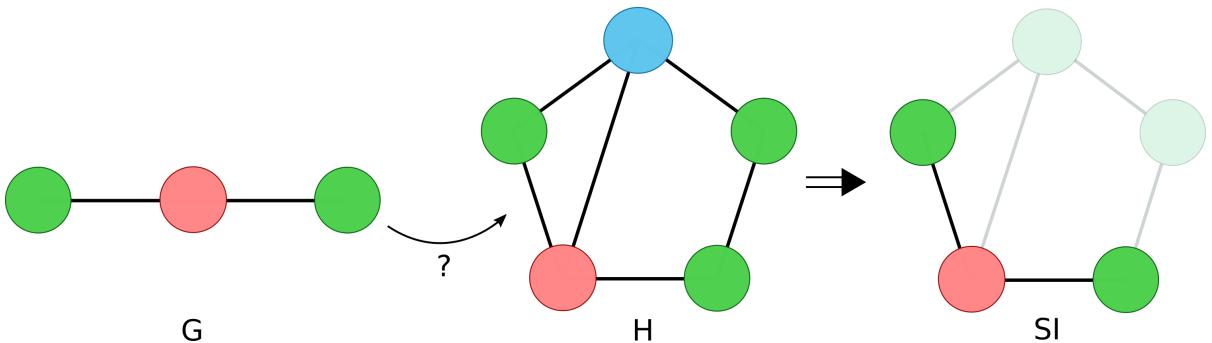


Figure 2.11: SI obtenu à partir de la recherche du graphe G dans H. Sur cet exemple, il n'y a qu'une seule possibilité mais si nous avions cherché une chaîne comme Vert-Rouge-Bleu, nous aurions obtenu deux SI différents.

Définissons désormais le problème d'isomorphisme de sous-graphe (Subgraph Isomorphism -SI-). Comme le nom l'indique, le problème SI est un problème de recherche d'un

graph dans une sous partie d'un autre. C'est-à-dire que G est sous-graphe de H lorsque tous les noeuds et toutes les arêtes de G sont retrouvées structurées de la même manière dans H (voir figure 2.11). Tout comme le MCS, le problème du SI a été prouvé NP-Complet [25].

Pour notre problème, nous pouvons à nouveau effectuer une réduction vers le problème SI, puis résoudre ce problème par des algorithmes déjà publiés. Nous souhaitons rechercher des monomères au sein de peptides, ce qui revient à chercher plusieurs fois un petit graphe atomique monomérique dans un gros graphe atomique peptidique.

Les méthodes de résolution actuelles sont toutes très similaires et dérivent d'une même technique de backtracking. La plupart de ces méthodes sont des résolutions exactes. Bien que NP-complet, ces algorithmes restent très praticables pour des graphes étiquetés, lorsqu'on ne recherche pas au sein de graphes répétitifs (qui sont constitués de multiples répétitions d'un même morceau). Voyons ensemble plusieurs algorithmes représentatifs.

2.3.2.2 L'algorithme de Ullman

L'algorithme de Ullman est l'un des premiers à avoir été proposé dans les années 70 [70]. Certaines de ses variantes sont toujours utilisées dans certaines librairies telles que Openbabel [47]. L'algorithme se base sur la matrice de compatibilité entre les graphes (voir figure 2.12). Une matrice de compatibilité est une matrice dont chaque ligne représente un noeud de G et chaque colonne un noeud de H . Soient g et h correspondant respectivement à un noeud du graphe G et du graphe H . Soient i et j les numéros des lignes et colonnes de la matrice correspondant à g et h . La case de coordonnées (i, j) contient un 1 si g et h ont une étiquette compatible et que l'arité de h est supérieure ou égale à l'arité de g . Pour obtenir un SI à partir de cette matrice, il faut retirer des 1 de la matrice jusqu'à en obtenir un seul 1 par ligne. A chaque étape, un noeud g de G est choisi pour continuer l'isomorphisme. Ce noeud correspond à une ligne i . Puis une colonne j est choisie de manière à ce que la case (i, j) de la matrice contienne un 1. Le couple (i, j) nous donne le couple (g, h) qui peut être ajouté à l'isomorphisme. Avant cet ajout, il faut également vérifier que tous les voisins de h correspondent bien aux voisins de g (condition non respectée aux étapes 2 et 5 de la partie recherche du schéma). Puisque nous ajoutons ce couple à l'isomorphisme, il faut remplacer tous les 1 de la ligne et la colonne choisies par des 0 (excepté pour la case (i, j)). À ce moment, si au moins une ligne a été vidée de tous ses 1, c'est que l'isomorphisme n'est pas possible et il faut revenir à l'étape précédente. L'algorithme continue récursivement jusqu'à obtenir un 1 par ligne ou ne trouver aucun isomorphisme. En choisissant bien les lignes et les colonnes par lesquelles l'algorithme commence (par exemple celles pour lesquelles il n'y pas de choix multiples), on arrive à obtenir des temps d'exécution très courts pour

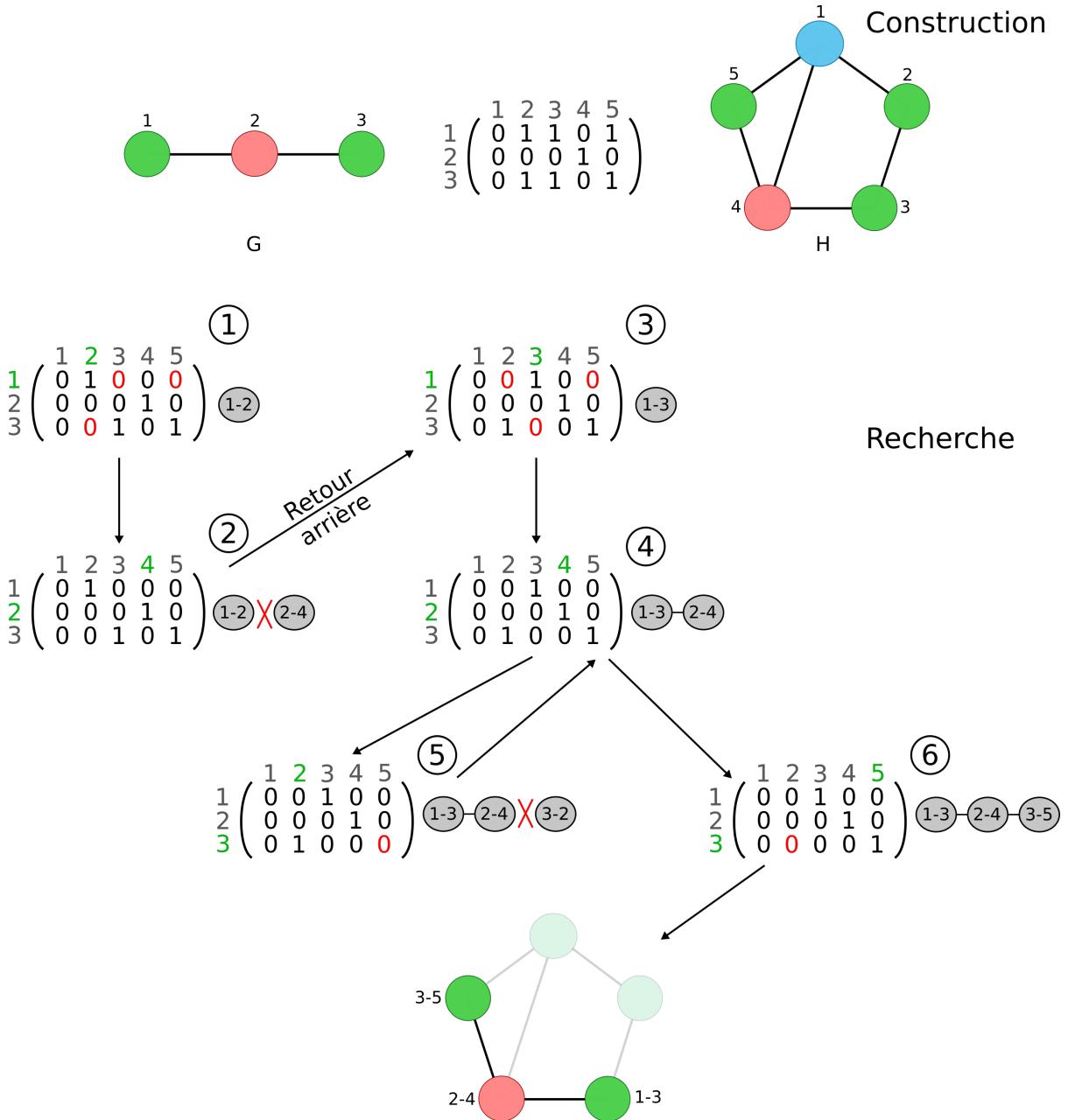


Figure 2.12: Recherche d'un isomorphisme de sous-graphe par l'algorithme d'Ullman. Dans la partie "construction", vous pouvez voir la matrice créée à partir des graphes G et H . Dans la partie "recherche" sont présentées les étapes successives de création d'un SI (voir le texte pour les détails). Si nous avions voulu obtenir tous les isomorphismes (et pas juste un seul), il aurait fallu continuer le backtracking jusqu'au bout.

la recherche d'un SI. Cependant, pour obtenir tous les SI, il faut parcourir tout l'arbre des possibles, ce qui ralentit l'algorithme.

2.3.2.3 VF2, un algorithme de backtracking classique

Tout comme pour le MCS, le SI peut être résolu par un backtracking classique par association de nœuds deux à deux. C'est la méthode majoritairement utilisée puisqu'elle fonctionne très bien en pratique. Contrairement à l'algorithme d'Ullman qui nécessite d'effectuer des opérations sur des matrices potentiellement très grandes, la méthode par backtracking classique n'a pas besoin de grands espaces mémoire. L'algorithme VF2 [15] utilise ce principe. Les SI se forment par ajouts successifs de couples de nœuds compatibles avec retour en arrière s'il n'est plus possible d'avoir un couple. VF2 est un algorithme de recherche de SI pour des graphes non orientés et non étiquetés.

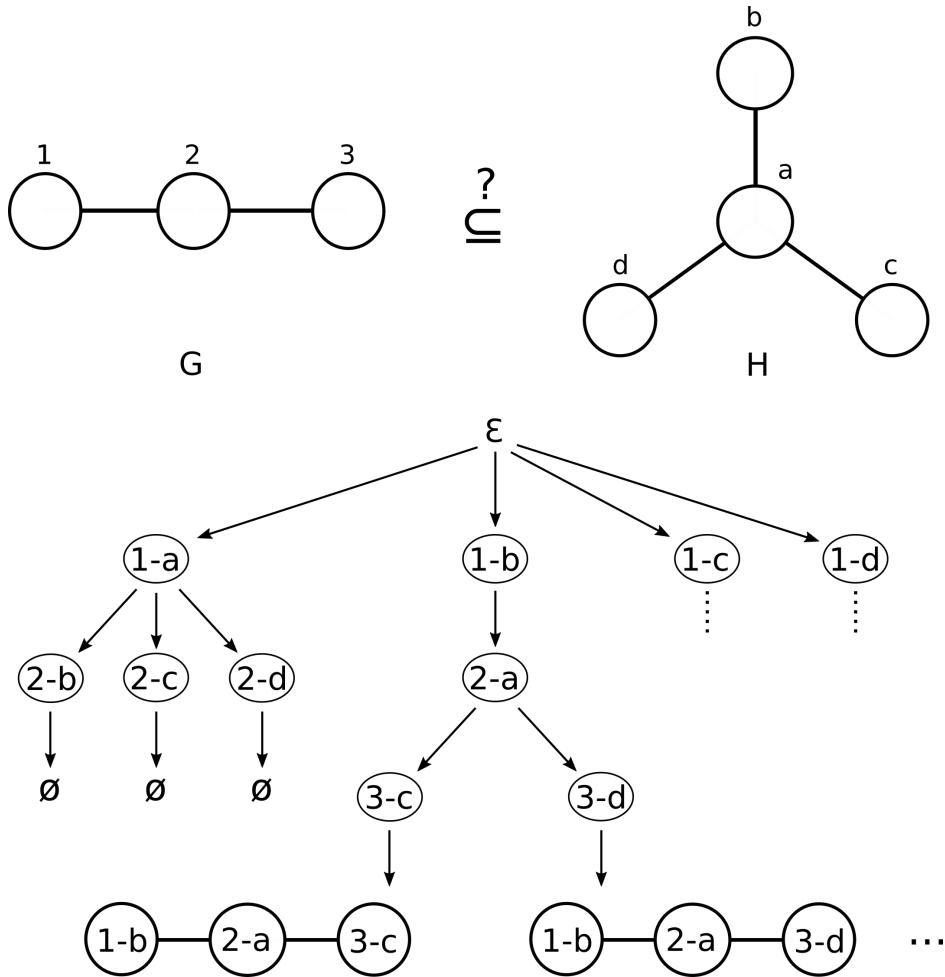


Figure 2.13: Backtracking de VF2 pour extraire l'ensemble de tous les SI possibles.

Expliquons cet algorithme en utilisant le schéma 2.13. À la première étape, nous nous trouvons avec un ensemble vide de nœuds couplés de G et H . Nous générerons donc toutes les possibilités de couples entre tous les nœuds des deux graphes (Il est à noter que l'algorithme

est décrit de cette façon dans l'article mais qu'il n'est nécessaire de générer que les couples à partir d'un seul nœud de G et de l'ensemble des nœuds de H .) Une fois ces couples créés, nous les parcourons tout en appelant récursivement la fonction. Dans l'exemple, le premier appel récursif est effectué après avoir assemblé '1' à 'a'. Cette voie est sans issue car quel que soit l'assemblage entre '2' et un nœud de H , plus aucun couple de voisins ne pourra être généré. Une fois toutes les tentatives échouées, l'algorithme remonte dans l'arbre et change l'association '1'-a' pour l'association '1'-b'. Récursivement, l'algorithme conclut à un isomorphisme (1,b);(2,a);(3,c).

En l'état, l'algorithme VF2 ne reconnaît qu'un seul et unique isomorphisme. On peut modifier simplement l'algorithme afin de récupérer tous les isomorphismes en ne s'arrêtant pas une fois que l'un d'eux est trouvé mais en l'enregistrant dans un ensemble puis en continuant l'exploration de l'arbre jusqu'au bout. L'arbre est entièrement exploré à chaque recherche, l'algorithme est donc exponentiel. Pour être plus précis, il est en $O(n^m)$, donc non calculable dans les pires des cas (n nombre de sommets de G et m de H).

À cause de sa complexité, cet algorithme est très pratique pour trouver un seul SI, mais pas pour les trouver tous. En pratique, il est utilisé au sein de grandes librairies logicielles de chimie comme CDK [67] pour faire des requêtes par structure sur des ensembles chimiques.

2.3.2.4 D'autres algorithmes de SI

Contrairement à la recherche de MCS, il n'existe pas beaucoup d'alternatives heuristiques pour le SI. Ceci peut s'expliquer par le fait que des algorithmes exacts et pratiques existent pour trouver les SI. De plus, les heuristiques de MCS permettent d'obtenir des résultats de SI en ajoutant des contraintes (Le SI étant un cas particulier du MCS). Dans tous les cas, ces heuristiques de SI sont des dérivés de la technique par backtracking [29]. Ces algorithmes essaient en supplément d'éviter l'exploration de toutes les branches en sélectionnant les plus "prometteuses". Par exemple, on préférera explorer des isomorphismes commençant par des paires de nœuds d'arité forte car ils sont un indice de "bon isomorphisme".

Il existe également de nombreux algorithmes pour des cas particuliers de SI. Par exemple, il est possible de résoudre le problème SI en temps polynomial pour des arbres [59], des graphes planaires [19, 21] ou des graphes de *treewidth* faible [28]. Dans tous les cas, il est possible de trouver au moins une molécule (pas forcément NRP) ne répondant pas à l'un de ces critères et c'est pour cela que nous avons préféré ne pas les utiliser.

2.3.3 Choisir l'algorithme de recherche de monomères

Pour pouvoir rechercher les monomères, nous devions choisir entre toutes les approches algorithmiques citées ci-dessus. Premièrement, lorsque nous recherchons un monomère dans un peptide, nous souhaitons connaître toutes les occurrences potentielles. Nous ne souhaitons donc pas utiliser d'approches heuristiques. Deuxièmement, le MCS ne paraît pas très pratique dans notre cas. En effet, les NRP regorgent de monomères proches les uns des autres (variation de un ou deux atomes/liaisons). Il n'est donc pas facile, lorsque l'on obtient un résultat, de savoir si la molécule trouvée est la bonne ou si c'est le monomère voisin à un atome de distance qui a été détecté. Enfin, en comparant les algorithmes de recherche exacte de solutions, ceux de résolution de SI sont plus adaptés et rapides en pratique. Au début, nous ne connaissions pas l'existence de l'algorithme VF2 et aucun des autres algorithmes ne nous convenait en temps de calcul. De notre côté, nous avons indépendamment redéveloppé un algorithme étendu de VF2 appliqué aux graphes étiquetés et suffisamment rapide pour pouvoir trouver tout SI quasi-instantanément. La section suivante donnera tous les détails concernant cet algorithme et son intégration à s2m.

2.4 Construction de Smiles2Monomers

2.4.1 Isomorphisme de sous-graphe appliqué à la recherche de monomères

Notre logiciel appelé Smiles2Monomers (s2m), doit débuter par la recherche d'un ensemble de monomères au sein d'un polymère cible. Comme nous l'avons vu dans l'état de l'art, nous pouvons transformer ce problème en un problème de graphe appelé isomorphisme de sous-graphe (SI). Dans cette partie, nous allons présenter en détail la variante que nous avons conçue, de l'algorithme de résolution de SI appelé VF2. Puis nous montrerons en quoi les étiquettes des noeuds du graphes influencent le temps de calcul afin d'en déduire une méthode de recherche efficace.

2.4.1.1 Notre algorithme pour les graphes étiquetés

Intéressons nous à l'application de VF2 aux graphes étiquetés. L'algorithme peut être modifié pour ne former que des paires de noeuds de même étiquette. Prenons à nouveau notre exemple précédent en étiquetant avec "vert" et "rose" les noeuds. Sur la figure 2.14, on peut voir la recherche d'un graphe G composé de 2 noeuds "vert" et un noeud "rose" au sein d'un graphe H composé d'un branchement supplémentaire contenant un noeud "rose". Sur

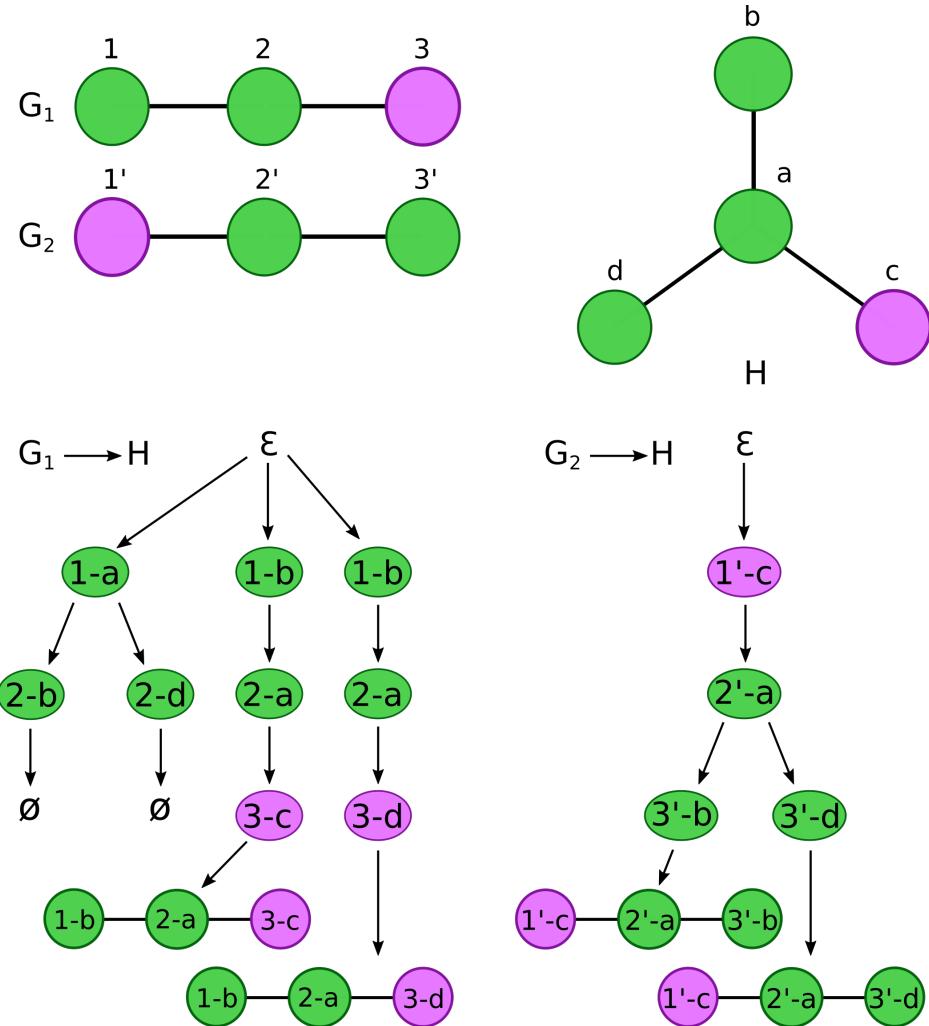


Figure 2.14: Modification de VF2 pour son application aux graphes étiquetés. On voit nettement sur cet exemple que l'ordre de recherche des étiquettes influence le temps d'exécution de l'algorithme.

ce graphique nous avons représenté deux cas de recherche selon l'ordre dans lequel nous prenons G . Nous pouvons remarquer que, dans les deux cas, la taille de l'arbre est bien plus faible que pour un graphe non étiqueté. Les étiquettes des noeuds font office de filtres de recherche et accélèrent fortement la recherche. On voit également que la recherche est d'autant plus rapide que l'on choisit prioritairement les noeuds avec des étiquettes de faible fréquence.

Là où l'algorithme général VF2 était en $O(n^m)$, la complexité est ici réduite dans le pire des cas à $O(k^n)$ où k est le comptage du nombre de noeuds de l'étiquette la moins filtrante. La taille de l'arbre de recherche est diminuée par la sélectivité des étiquettes de noeuds. Pour optimiser la recherche, nous devons donc étudier la façon dont nous représen-

tons les molécules afin d'avoir la sélectivité la plus grande possible.

2.4.1.2 Sélectivité d'un nœud

Premièrement, attardons nous sur la représentation la plus basique possible d'un graphe chimique (atome=nœud, liaison=arête). Dans cette représentation, nous pouvons considérer chaque étiquette comme étant le nom de l'atome du nœud (Voir figure 2.15, première partie). Bien entendu, les hydrogènes sont les atomes les plus représentés et ne filtrent pas les recherches par eux-mêmes. Cependant, ils contiennent une information supplémentaire sur l'atome auquel ils sont liés. Ainsi, trouver un CH_3 ou un CH donne une information différente sur la topologie. En prenant en compte la remarque précédente, nous avons créé une seconde représentation qui compresse les hydrogènes à l'intérieur des nœuds représentant les atomes plus lourds. Ainsi, plusieurs étiquettes différentes peuvent désormais être créées pour un même atome et ce, en fonction de ses compagnons hydrogènes (voir figure 2.15, seconde partie). Par exemple NH_2 , NH , N sont les trois étiquettes variants depuis l'ancienne étiquette unique N qui était accompagnée de voisins hydrogènes. Les étiquettes sont plus nombreuses et leur fréquence à la fois moins élevée et plus homogène, ce qui les rend plus sélectives.

Globalement, au plus le nombre d'étiquettes est élevé et au plus les fréquences de ces étiquettes sont homogènes, au moins l'algorithme VF2 aura de difficultés à effectuer un isomorphisme. En effet, la lenteur qui peut venir de cet algorithme est due aux nombreux chemins qu'il peut prendre lors de l'exploration.

Cherchons encore à augmenter la sélectivité des étiquettes en augmentant leur nombre. A partir d'un graphe G , il est possible de représenter ses relations d'adjacence au sein d'un second graphe nommé L . Pour cela, deux nœuds voisins de G sont représentés dans L comme un seul nœud (voir line-graph2 dans la figure 2.15). Chaque ancienne arête devient un nœud et chaque ancien nœud devient une multitude d'arêtes (une clique entre tous les nouveaux nœuds représentant les anciennes arêtes de cet ancien nœud). Nous étendons le raisonnement pour les étiquettes en disant que l'étiquette d'un nœud du nouveau graphe est la composition des étiquettes des anciens nœuds ainsi que de l'arité entre ces deux nœuds. Dans notre cas, les étiquettes sont mises dans l'ordre lexicographique pour ne pas avoir deux étiquettes possibles à partir des étiquettes d'origine. Un tel type de graphe est appelé *line graph* [48]. L'inclusion de l'arité du lien dans les étiquettes est un très gros avantage de cette représentation. La fonction de recherche n'aura donc plus du tout à s'attarder sur le degré des arêtes puisque le tout sera reporté sur la fonction de matching d'étiquettes (voir figure 2.16). De plus, transformer les graphes de cette manière nous permet de répondre à

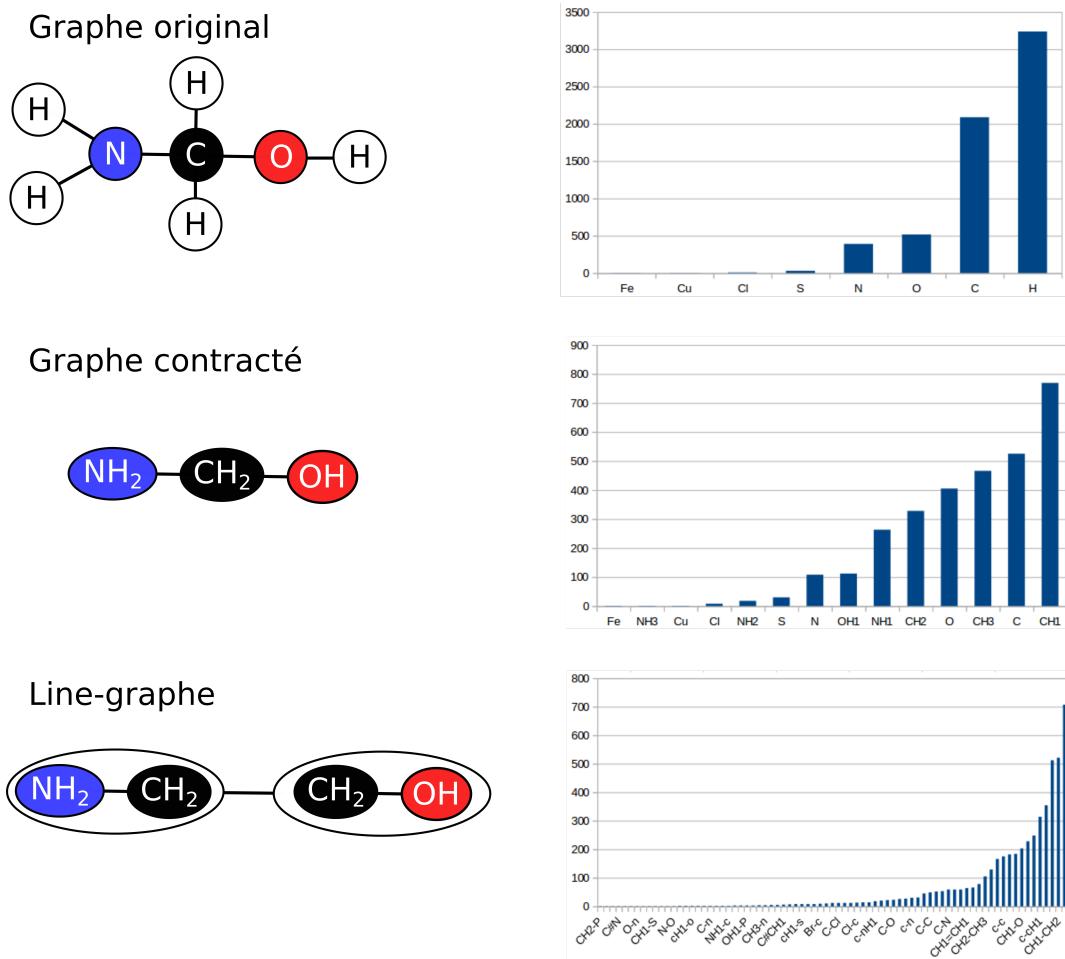


Figure 2.15: Triple représentation de la même molécule accompagnée des fréquences de chacune des étiquettes possibles. Ces fréquences ont été calculées sur un extrait de 200 molécules issues de la base de données Norine.

la volonté d'augmenter le nombre d'étiquettes. Là où la représentation précédente pouvait avoir n étiquettes différentes, celle-ci pourra en générer plus de n^2 ($3n^2$ si on compte tous les degrés des liaisons entre atomes).

Le raisonnement des *line graphs* peut être étendu récursivement. On peut ainsi créer des *line-graphs* de *line-graphs*. Cependant, dans notre cas, cela pose plusieurs problèmes de répéter cette opération. La répétition de l'opération de transformation diminuera la taille de certains graphes jusqu'à ce qu'ils disparaissent totalement. Dans d'autres graphes, le nombre de noeuds va grandir avec la répétition et ainsi augmenter le nombre de comparaisons nécessaires à la résolution d'un SI. Pour ces raisons, nous avons choisi de n'effectuer qu'une seule étape de *line-graph*.

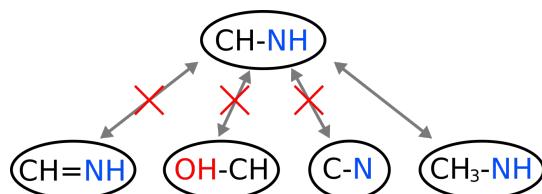


Figure 2.16: Exemple de fonctionnement de la fonction de matching d'étiquettes. Le premier nœud ne correspond pas au nœud cible car l'arité du lien contenue dans l'étiquette est différente. Le deuxième ne correspond pas car les atomes des étiquettes ne correspondent pas. Le troisième ne correspond pas car l'étiquette ne contient pas suffisamment d'hydrogènes liés aux atomes principaux. Enfin, le dernier remplit les 3 critères précédemment cités et la correspondance est possible.

2.4.1.3 Indexation des monomères

Connaissant la sélectivité pour une étiquette, nous cherchons désormais à savoir l'ordre dans lequel il faut rechercher un motif afin de minimiser le temps de recherche. Nous appellerons **motif** le graphe compressé que nous allons vouloir retrouver dans un plus grand graphe. Définissons également une **chaîne** comme étant un motif plus un **ordre** dans lequel parcourir ce motif. Ainsi, on peut avoir plusieurs chaînes représentant le même motif mais avec des ordres de parcours différents. Ce que nous appellerons **indexation** sera donc la phase de création d'une chaîne pour représenter chaque motif d'intérêt en essayant de minimiser le temps que prendra la recherche de cette chaîne. Pour la recherche de monomères dans des peptides non ribosomiques, la phase d'indexation revient donc à choisir, pour chaque monomère, l'ordre dans lequel nous allons rechercher les nœuds du graphe le représentant. Ainsi, on peut voir que quelle que soit la chaîne représentant le graphe, le même motif sera toujours recherché. La seule différence sera la rapidité d'exécution de la recherche. Les deux sections suivantes s'attarderont à la recherche de chaînes performantes (*i.e* les plus rapides) lors la recherche d'un motif.

2.4.1.4 Indexation gloutonne

Une chaîne efficace est donc une chaîne rapide. Déterminons quel(s) critère(s) simple(s) nous permettent de supposer qu'une chaîne est efficace. Avec ce ou ces critères, nous pourrons ensuite déterminer un moyen d'ordonner les nœuds de manière gloutonne. Partons de l'exemple présenté sur la figure 2.17. À gauche est dessiné un motif simple suivi de deux chaînes pouvant le représenter (lecture de l'ordre des chaînes de gauche à droite). Pour la première chaîne, en partant du motif ne contenant qu'un “vert”, on obtiendra 3

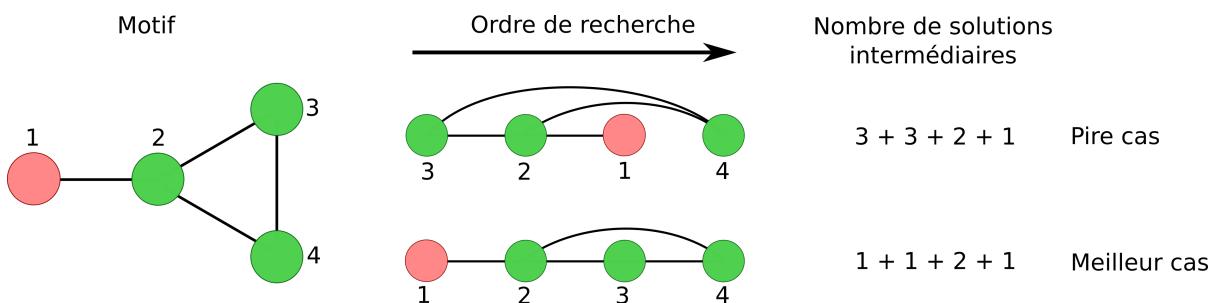


Figure 2.17: Deux exemples de chaîne pour un motif simple. Le nombre de solutions intermédiaires représente les nombres successifs d’isomorphismes de sous-

isomorphismes. En étendant ces chaînes à deux “vert” voisins, on conserve toujours trois isomorphismes puis deux en ajoutant “rouge” et enfin un seul avec le dernier “vert”. Au total, on aura étendu 9 isomorphismes durant la recherche. En regardant maintenant la seconde chaîne, on se rend compte que le nombre de ces isomorphismes intermédiaires est bien plus faible grâce à la sélectivité de l’étiquette “rouge” recherchée en tout premier (5 isomorphismes intermédiaires sont étendus). La différence décisive entre les deux chaînes est donc le moment où le nœud avec une étiquette sélective est recherché. Effectivement, si le nœud “rouge” est recherché dès le début, la solution est “ancrée” sur une base solide et ne risque a priori pas de créer trop de solutions intermédiaires qui devront toutes être analysées pour passer à l’étape suivante. On peut dire qu’un nœud est plus sélectif qu’un autre à partir du moment où sa fréquence d’apparition dans les molécules étudiées est moins élevée. Il sera souvent plus intéressant de commencer les recherches des motifs par des nœuds dont l’étiquette est peu fréquente.

Étiquettes fréquentes

Essayons d’estimer ces fréquences afin d’éviter les cas qui pourraient être critiques. Pour ce faire, nous pouvons constituer un ensemble représentatif du type de molécules que nous allons analyser. Dans notre cas, nous cherchons à étudier des peptides non ribosomiques. La base d’apprentissage des fréquences sera donc constituée de NRP dont la composition atomique est déjà connue. Le raisonnement est extensible à n’importe quel type de molécules et le fait d’avoir un jeu d’apprentissage peu ou pas représentatif n’influencera pas le résultat mais uniquement le temps de calcul. Une fois cette base créée, nous pouvons estimer les fréquences d’apparition de chacune des étiquettes présentes dans les monomères en effectuant une recherche exhaustive (Figure 2.18).

Un algorithme glouton naïf vient tout de suite en tête. Il est possible de choisir un à un les nœuds que l’on souhaite ajouter au motif en prenant toujours celui avec l’étiquette la

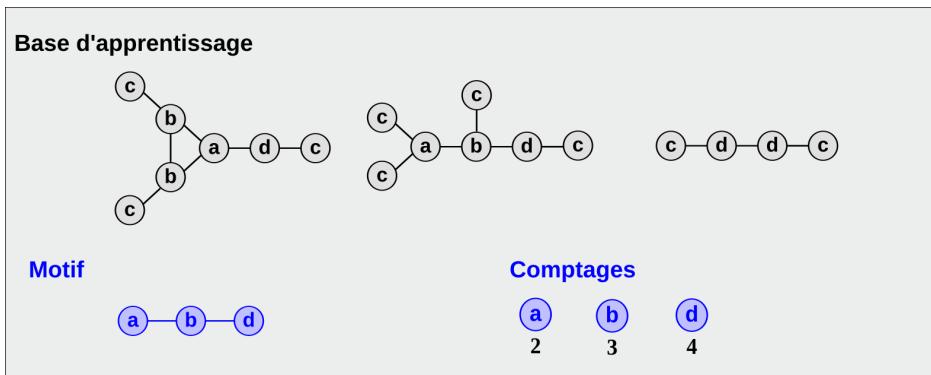


Figure 2.18: En haut de l'image est représentée la base d'apprentissage puis en bas, le motif qui doit être indexé ainsi que les fréquences d'apparition de chacun de ses nœuds dans la base d'apprentissage

moins fréquente. Il est nécessaire de préciser que la recherche des nœuds voisins peut être faite en maintenant un ensemble des voisins. L'algorithme glouton naïf est donc linéaire par rapport à la taille du graphe. À l'exécution, la création de chaînes par cette méthode est instantanée ($\ll 0.1s$). Toutefois, rien ne garantit la qualité des chaînes obtenues. Prenons l'exemple de la figure 2.18. Sur cet exemple, le motif à indexer est a-b-d. Les comptages des différentes étiquettes nécessaires sont également présents sur cette même figure. La chaîne qui sera générée commence forcément par le nœud étiqueté *a* (fréquence la plus faible dans le jeu d'apprentissage) suivi de *b* puis enfin *d* (le tout est résumé sur la figure 2.19).

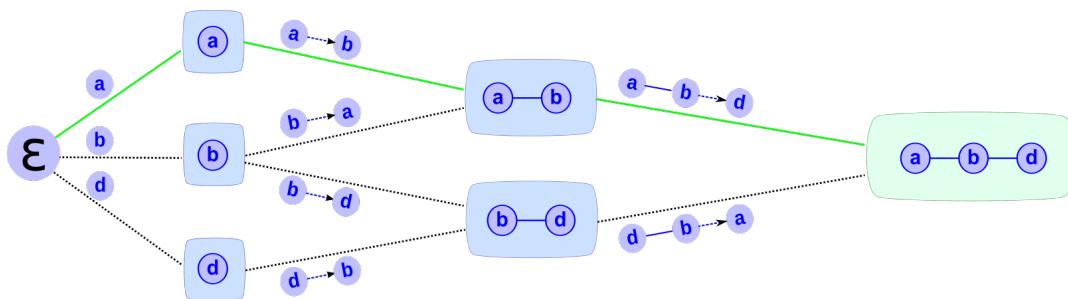


Figure 2.19: Ensemble des chaînes générées à partir du motif a-b-d avec en vert le chemin pris par l'algorithme glouton.

2.4.1.5 Indexation Markovienne

En utilisant la méthode gloutonne, nous obtenons rapidement les chaînes de tous les motifs que nous souhaitons indexer. Cependant, dans certains cas, une chaîne générée de cette

manière peut ne pas être optimale. Dès le tout début de la chaîne, il est possible que l'algorithme fasse des choix localement optimaux qui vont ensuite mener à un enlisement global. Nous allons nous appuyer sur les figures 2.18 et 2.19 pour montrer que même sur un si petit exemple, l'algorithme glouton n'a pas choisi le meilleur chemin.

Afin d'évaluer la qualité d'une chaîne, nous allons calculer son temps d'exécution sur notre base d'apprentissage. Partons de la définition d'une chaîne : une chaîne est composée d'une succession de noeuds. Ces noeuds vont, lors de l'isomorphisme, être recherchés dans l'ordre donné par la chaîne. Pour rechercher une chaîne de taille n ($c_n = p_0 \rightarrow p_1 \rightarrow \dots \rightarrow p_n$), il faut donc n extensions successives. On peut donc écrire que le temps de recherche d'une chaîne est le temps cumulé de recherche de ses extensions :

$$T(c_n) = T(p_0 \rightarrow p_1 \rightarrow \dots \rightarrow p_n) = \sum_{i=1}^n T(p_{i-1} \rightarrow p_i) \quad (2.3)$$

Où $T()$ est la fonction représentant le temps de recherche d'une chaîne, p_i le motif de taille i et $p_{i-1} \rightarrow p_i$ la dernière extension d'une chaîne c_i représentée par le passage du motif p_{i-1} au motif p_i . Il est à noter que rechercher une chaîne de taille 1 revient à étendre un motif vide. C'est à dire qu'il est nécessaire de parcourir l'intégralité des noeuds de la cible dans laquelle on recherche le motif et ce quel que soit le motif. Sur notre base d'apprentissage il faudra donc toujours tester tous les noeuds de tous les peptides d'apprentissage pour la première extension.

Lors d'une recherche d'une chaîne c_{n-1} , il est possible de trouver des isomorphismes sur la cible (potentiellement partiellement recouvrant). Afin de calculer le temps de recherche de c_n nous devons prendre en compte la quantité de sous-isomorphismes précédemment trouvés. Le calcul du temps de recherche de c_n correspondra donc au temps de calcul de c_{n-1} auquel on vient ajouter le temps de recherche de la dernière extension sur chacun des isomorphismes de taille $n - 1$. Il faut également tenir compte du nombre d'extensions qui vont être essayées. Par exemple, si l'on cherche à étendre un isomorphisme précédent en suivant un lien qui est enraciné sur un NH , il ne reste qu'une liaison à explorer (le N ayant déjà une liaison avec son H et une liaison avec le reste du motif déjà recherché). En reprenant la même logique pour un seul carbone, il resterait donc trois autres liaisons à suivre. Nous appellerons cette quantité l'*arité restante*. On peut donc écrire :

$$T(p_{n-1} \rightarrow p_n) = n_{p_{n-1}} \times a_{p_{n-1} \rightarrow p_n} \times t \quad (2.4)$$

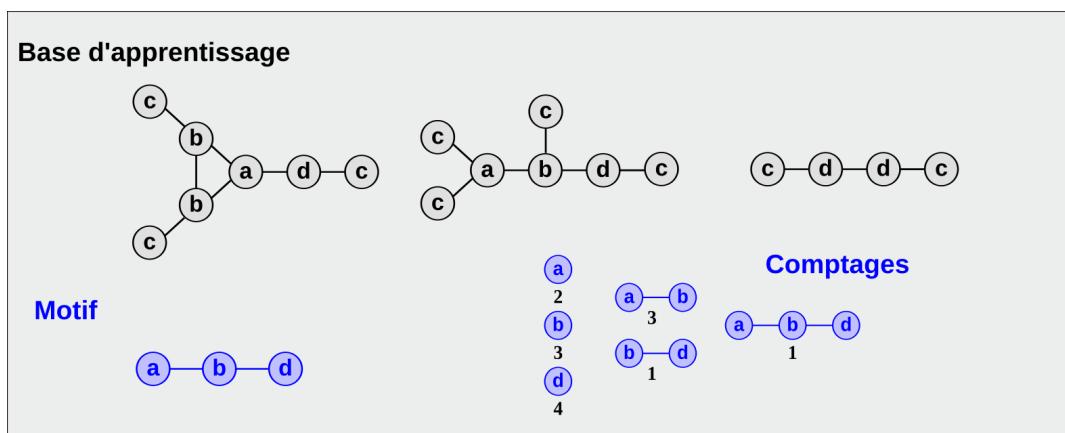
où $n_{p_{n-1}}$ est le nombre d'isomorphismes trouvés pour le motif p_{n-1} , $a_{p_{n-1} \rightarrow p_n}$ est

l'arité restante de l'atome depuis lequel on étend le motif p_{n-1} vers le motif p_n et t le temps nécessaire pour une comparaison de deux étiquettes (ce temps peut être considéré constant)

Combinons les deux équations précédentes, pour obtenir une formule globale :

$$T(c_n) \propto^t N + \sum_{i=2}^n (n_{p_{i-1}} \times a_{p_{i-1} \rightarrow p_i}) \quad (2.5)$$

où N est le temps de calcul pour les chaînes de taille 1. Ce temps ne varie pas car il est nécessaire de parcourir tout le graphe.



Filtrages :

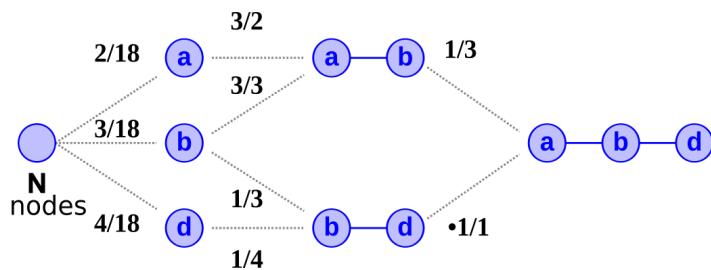


Figure 2.20: Reprise de la base d'apprentissage présentée en figure 2.18 pour définir le filtrage. Les comptages initiaux des fréquences de chacun des noeuds ont été étendus aux comptages de l'ensemble des sous-motifs possibles. Le taux de filtrage d'un ajout de noeud dans un motif est alors défini comme le nombre de détections dans la base du motif étendu divisé par le nombre de détections du motif initial. Ainsi, pour passer d'un motif a à un motif a-b, le taux de filtrage est de 3/2.

En utilisant les comptages faits sur la base d'apprentissage (voir 2.20) ainsi que le calcul de l'arité restante de chaque noeud à chaque étape, nous allons pouvoir estimer un temps d'exécution de la recherche d'une chaîne. Ce temps est un coût exact de la recherche

sur la base d'apprentissage. En normalisant cette valeur par le nombre de nœuds de la base, nous pouvons ainsi obtenir le coût de recherche par nœud et ainsi estimer le temps de calcul, quelle que soit la taille de la, ou des, molécule(s) à analyser.

$$T(c_n) \propto^t 1 + \sum_{i=2}^n \left(\frac{n_{p_{i-1}}}{N} \times a_{p_{i-1} \rightarrow p_i} \right) \quad (2.6)$$

Redéfinissons la notation $\frac{n_{p_{i-1}}}{N}$ qui correspond à un taux de filtrage d'un motif de taille $i - 1$. Appelons ce taux F_{i-1} . Cette notation peut à son tour être transformée en produit des filtrages de chaque extension d'une chaîne. Les valeurs des filtrages estimés s'obtiennent immédiatement à partir des comptages dans la base d'apprentissage (voir l'exemple en figure 2.20). Il est à noter que le taux de filtrage peut être supérieur à 1 dans le cas où la recherche obtient plus d'isomorphismes en cours de création à la suite de l'ajout du nœud.

$$\frac{n_{p_{i-1}}}{N} = F_{i-1} = \prod_{i=0}^{i-1} f_i \quad (2.7)$$

$$T(c_n) \propto^t 1 + \sum_{i=2}^n a_{p_{i-1} \rightarrow p_i} \prod_{j=0}^{i-1} f_j \quad (2.8)$$

Le temps de recherche optimal d'un motif est donc le temps de recherche de la chaîne qui minimise le coût. Donc :

$$T(p_n) = \min_{\forall c_n} T(c_n) \quad (2.9)$$

Appliquons maintenant cette formule de calcul du temps sur notre exemple pour trouver la meilleure chaîne possible par rapport à la base d'apprentissage. Pour le motif de l'exemple de la figure 2.20, nous pouvons nommer les différentes chaînes possibles de α jusqu'à δ .

$$\begin{aligned} c &= p_1 \rightarrow p_2 \rightarrow p_3 \\ c_\alpha &= a \rightarrow a-b \rightarrow a-b-d \\ c_\beta &= b \rightarrow a-b \rightarrow a-b-d \\ c_\gamma &= b \rightarrow b-d \rightarrow a-b-d \\ c_\delta &= d \rightarrow b-d \rightarrow a-b-d \end{aligned}$$

On peut appliquer la formule sur chacune des chaînes possibles et trouver la chaîne

optimale qui minimise la fonction de coût. Développons l'exemple pour la chaîne c_α .

$$\begin{aligned}
 c_\alpha &= \epsilon \rightarrow a \rightarrow a-b \rightarrow a-b-d \\
 T(c_\alpha) &= T(\epsilon \rightarrow a) + T(a \rightarrow a-b) + T(a-b \rightarrow a-b-d) \\
 &\propto_N 1 + 3 \times \frac{2}{18} + (3-1) \times \frac{2}{18} \times \frac{3}{2} \\
 &= 1 + \frac{1}{3} + \frac{1}{3} = \frac{5}{3}
 \end{aligned}$$

En appliquant cette formule sur toutes les chaînes, on peut se rendre compte que certains de nos calculs sont inutiles. Prenons l'exemple des chaînes c_α et c_β . Le dernier calcul de la chaîne c_β est inutile. Lors du calcul des deux chaînes, on aurait pu se rendre compte que l'on pouvait déjà décider de la meilleure chaîne à partir du résultat de la sous chaîne $a-b$. Ceci nous amène à transformer le calcul sous forme de programmation dynamique dont les formules de récurrence sont les suivantes.

$$T(\epsilon \rightarrow p_1) = 1 \quad (2.10)$$

$$T(p_n) = \min_{\forall p'_{n-1} \subset p_n} (T(p'_{n-1}) + T(p'_{n-1} \rightarrow p_n)) \quad (2.11)$$

où la fonction de minimisation s'applique sur l'ensemble des p'_{n-1} où p'_{n-1} est un motif connexe de taille $n-1$ inclus dans p_n . Pour la recherche de motif dans l'exemple, on obtient le chemin c_δ minimisant le coût. On peut voir le détail de la recherche de ce motif sur la figure 2.21.

Cette méthode exacte nous permet de garantir la qualité de nos chaînes sur la base d'apprentissage mais cela se paye logiquement en temps de pré-calcul. En effet, même par programmation dynamique, le temps de pré-calcul des chaînes est très élevé si les motifs sont composés de nombreux branchements et d'une grande diversité de noeuds. Calculons la complexité, en considérant que chaque étiquette est unique, sur les deux cas extrêmes : une ligne et une clique.

Une ligne représente le motif le plus simple. Pour une ligne de noeuds de taille n , on peut trouver un motif de taille n puis deux de taille $n-1$ puis 3 de taille $n-2$, ... Pour calculer le coût du motif de taille n , il nous faut calculer récursivement tous les motifs de taille inférieure. Dans le meilleur des cas la programmation dynamique sera donc de complexité de l'ordre de n^2 .

Le pire motif qui peut être défini est une clique. Dans le cas d'une clique de taille n , pour calculer le coût du motif, il faudra calculer $n-1$ parmi n valeurs puis $n-2$ parmi

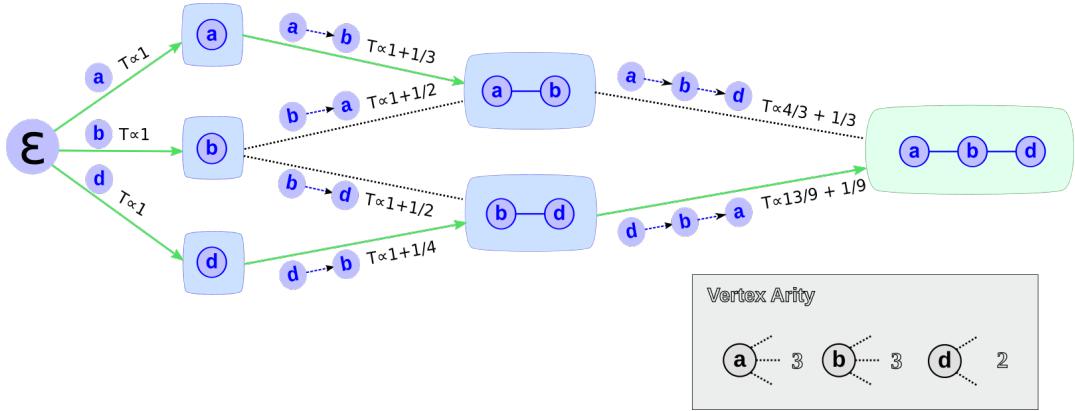


Figure 2.21: En vert, les chemins pris par la programmation dynamique pour finalement obtenir le chemin optimal vers le motif a-b-d. Il est également possible de rechercher le chemin optimal en utilisant l'algorithme de Dijkstra mais la mise en place des structures de données nécessaires ralentirait le calcul pour de petites données. Comme nous le verrons en section 2.5.3.1, nous ne calculerons jamais de grandes chaînes exactes sur nos données (chaînes exactes de taille 3 en général).

n, \dots Dans ce cas, la complexité sera de l'ordre de 2^n .

La redondance des étiquettes permet quant à elle de couper certaines branches lors de l'indexation et ainsi de diminuer un peu la complexité en pratique. Cependant, l'ordre de grandeur n'en est pas modifié.

$$Cplx_{chaine} = n + n - 1 + \dots + 2 + 1 = \sum_{i=1}^n i = \frac{n(n+1)}{2} = O(n^2) \quad (2.12)$$

$$Cplx_{clique} = \binom{n}{n} + \binom{n}{n-1} + \dots + \binom{n}{2} + \binom{n}{1} = \sum_{k=1}^n \binom{n}{k} = O(2^n) \quad (2.13)$$

2.4.1.6 Indexation hybride

Dans un exemple de la figure 2.21, nous avons montré que le début d'une chaîne est très critique lors de sa recherche dans un graphe. L'idée est donc de générer la chaîne par un mélange des deux techniques précédentes. Les débuts de la chaîne seront optimisés de manière exacte jusqu'à une taille k (préalablement définie par l'utilisateur), puis le reste des extensions se fera en utilisant l'algorithme glouton. Cette découpe des chaînes en deux parties garantit une recherche très rapide d'un motif sur la base d'apprentissage tout en limitant fortement le temps de pré-calcul nécessaire pour la construction de l'index.

2.4.2 Des monomères aux résidus

2.4.2.1 Résidus

L'algorithme et les indexés présentés dans la section précédente permettent de retrouver rapidement toutes les occurrences d'un motif au sein d'une molécule cible. Jusqu'à présent, nous avons pris l'hypothèse que nous recherchions des monomères dans des peptides. Ceci n'est en fait pas applicable directement. En effet, on ne retrouve jamais un monomère en entier au sein d'un polymère. Lorsque les monomères sont liés entre eux pour former le polymère, chacun perd une partie de ses atomes dans ces liaisons. Nous appellerons les molécules incluses (monomères sans les atomes de liaison) des **résidus**.

Prenons l'exemple de la Cystéine pour bien comprendre (voir figure 2.22). La cystéine est présente dans 47 peptides de la base de données NORINE. Dans certains peptides comme ACV, la cystéine est incluse en effectuant deux liaisons peptidiques avec ses voisins. D'un côté, elle perd les atomes *OH* de son groupement *COOH* pour aller se lier à un groupe *NH₂* qui perd un de ses *H*. Le tout libère un *H₂O* pendant la réaction. De l'autre côté, elle perd un *H* de son groupement *NH₂* pour effectuer la même opération inversée. Au total, la cystéine présente dans ce peptide a donc perdu trois atomes sur deux sites distincts. Dans d'autres peptides comme la malformin A1, la cystéine effectue une troisième liaison au niveau de son atome de soufre en perdant un atome d'hydrogène pour se lier avec un autre atome de soufre et ainsi réaliser un pont disulfure. Dans ce cas, on remarque que non seulement le monomère n'est jamais inclus en entier mais en plus, il peut être inclus dans un polymère de différentes manières. Il est donc nécessaire de connaître les différents résidus possibles pour chaque monomère afin de pouvoir les rechercher en utilisant l'algorithme de recherche exact décrit précédemment.

2.4.2.2 Familles de résidus

Différents résidus d'un même monomère peuvent être créés selon les contextes. Pour connaître les différents résidus possibles pour chaque monomère nous devons connaître les règles qui régissent les liaisons. Par exemple, pour la liaison peptidique précédemment citée, nous pouvons déduire deux règles. Si un *COOH* est détecté alors le monomère peut potentiellement perdre son *OH*. De la même manière, pour la seconde moitié de la liaison, le monomère peut perdre un *H* pour un *NH₂* détecté. En regardant précisément les liaisons qui apparaissent pour un type de polymère, nous pouvons créer une liste de règles de perte d'atomes. En appliquant ces règles récursivement sur un monomère, nous pouvons générer tous ses résidus candidats. Notons que certaines règles comme celles du pont disul-

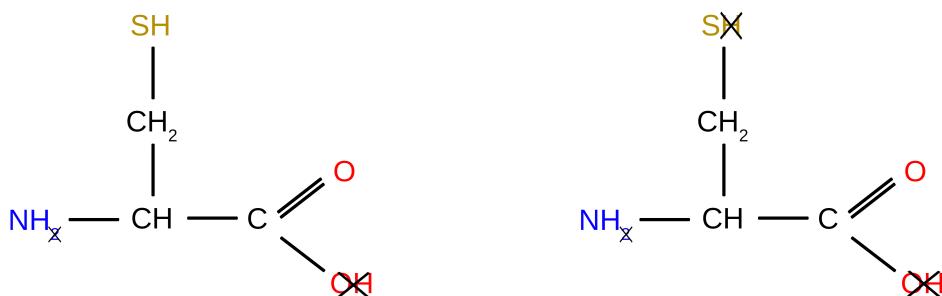


Figure 2.22: Deux résidus différents présents dans deux peptides différents tous deux obtenus à partir de la Cystéine

fure décrite précédemment, ne s’appliquent qu’en présence d’une autre liaison. Il n’est donc a priori pas nécessaire de générer le résidu qui contient uniquement cette liaison, car nous n’aurons normalement pas de résidu uniquement impliqué dans une liaison sulfure. Il est à noter que, moins une règle sera précise, plus le nombre de résidus générés sera important. Par exemple, imaginons une règle qui nous dirait dans un cadre de chimie organique, que “pour tout carbone lié à au moins un hydrogène, l’atome d’hydrogène peut être perdu pour former une liaison”. Si n est le nombre d’atomes de carbone dans la molécule (n représente une grande proportion d’atomes dans une molécule organique), alors on peut générer un nombre de résidus de l’ordre de n^2 .

Comme présenté dans la figure 2.23, les différents résidus générés peuvent être représentés sous la forme d’un Graphe Orienté Acyclique (GOA). Le monomère de base (en bas sur la figure) ainsi que chacun des résidus est un nœud du graphe et chaque arc est un lien de parenté entre nœuds. Un arc sera par exemple créé entre un résidu auquel on a appliqué une seule règle et le monomère. Nous appellerons **famille** cette représentation des résidus sous forme de GOA.

Le nœud le plus éloigné du monomère (en haut sur la figure) sera le nœud qui aura subi le plus de modifications. Nous appellerons le résidu de ce nœud le **résidu racine**. Ce résidu a un intérêt tout particulier pour la recherche du monomère. En effet, si le résidu racine n’est pas trouvé alors qu’il est celui qui possède le moins d’atomes dans la famille, cela signifie qu’aucun membre de la famille ne pourra être trouvé. On peut même aller plus loin en étendant le raisonnement à tous les résidus. Si un résidu de niveau n n’est pas trouvé, alors il est inutile de rechercher ses résidus parents de niveau $n - 1$. Il faut également remarquer qu’il est inutile de chercher un résidu parent de niveau $n - 1$ si aucun

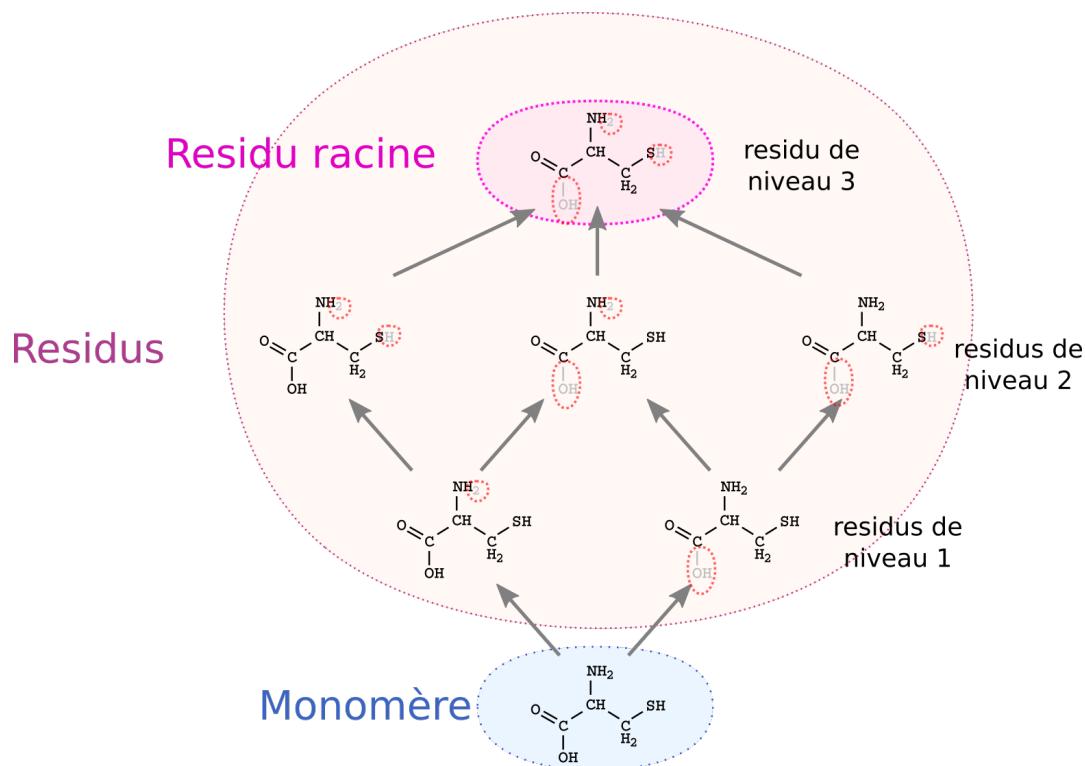


Figure 2.23: Représentation de la construction de la famille des résidus de la cystéine.

de ses enfants de niveau n n'ont été trouvés.

En connaissant la structure de la famille des résidus ainsi qu'un ordre naturel de recherche pour passer d'un résidu à l'autre, nous pouvons déduire une structure globale de recherche. Découle donc de cette structure de famille de résidus, un ordre de recherche naturel depuis le résidu racine jusqu'aux résidus de niveau 1 (inutile de rechercher le monomère car il n'est jamais inclus tel quel). De plus, il n'est jamais nécessaire de recalculer complètement un isomorphisme une fois que la racine est trouvée. Pour passer d'un résidu r à l'un de ses parents p il suffit de rechercher à étendre l'isomorphisme de r avec les atomes que nous avions considérés perdus pour passer de p à r lors de la construction de la famille. De plus, le fait de ne pas tout recalculer à chaque étape nous autorise à ne calculer l'index que pour les résidus racine. Le reste des résidus n'est indexé que sur les extensions à effectuer depuis leurs fils.

L'algorithme de recherche des résidus ressemble fortement à un algorithme de recherche en largeur. Nous commençons par rechercher le résidu racine. Si celui-ci est trouvé, alors tous ses résidus fils sont recherchés. Sur l'exemple de la figure 2.24, le résidu racine de la cystéine est trouvé ; les trois résidus de niveau 2 sont alors recherchés. Récursivement,

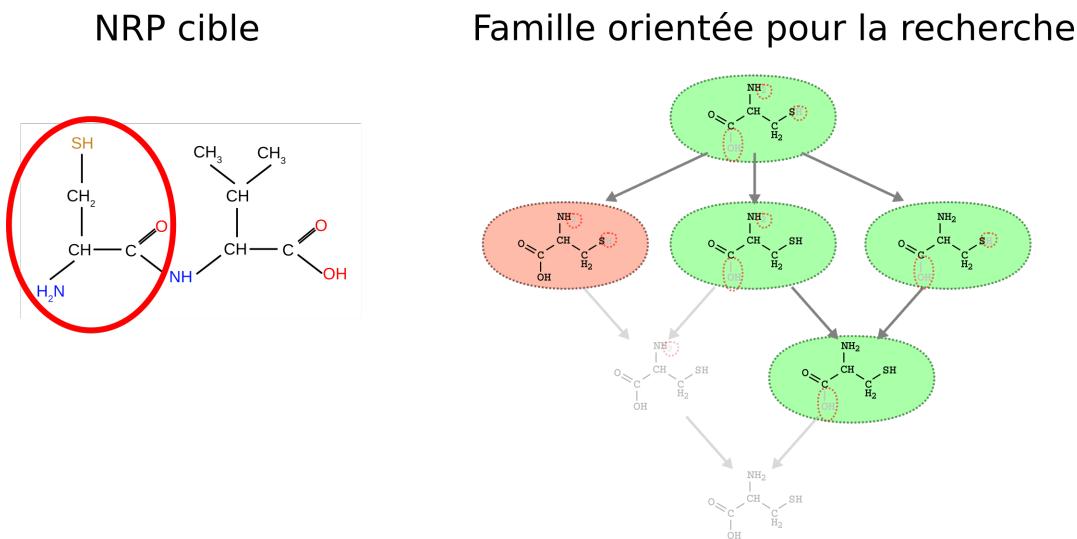


Figure 2.24: Représentation de la recherche de la cystéine dans un di-peptide, en utilisant sa famille de résidus. En vert, les résidus trouvés dans le NRP cible, en rouge ceux non trouvés et en transparent, les non cherchés.

lorsqu'un résidu est trouvé, tous ses fils sont recherchés. Ainsi, la recherche de la cystéine dans le peptide de l'exemple aboutit sur la découverte de 3 isomorphismes de monomères différents. Ces isomorphismes sont tous situés au même endroit et c'est la phase de pavage qui choisira celui à garder.

2.4.3 Pavage de monomères

2.4.3.1 Pavage : définition

La méthode de recherche des isomorphismes de sous-graphe présentée en section 2.3.2, trouve toutes les occurrences de chacun des monomères dans un peptide cible.

Puisque nous connaissons les atomes contenus dans l'ensemble des isomorphismes découverts, nous pouvons également connaître chacun des isomorphismes découverts et ce, par atome. L'étape de pavage consiste à choisir un sous ensemble des isomorphismes trouvés, de manière à maximiser le nombre d'atomes couverts. Il faut également ajouter la contrainte qu'aucun atome ne peut être couvert par deux isomorphismes dans la solution finale. Chaque atome provenant d'un unique monomère, il n'est pas possible d'autoriser des isomorphismes chevauchants.

Faisons une analogie avec un puzzle. La phase d'isomorphisme crée de nombreuses pièces de puzzle. Une reconstruction complète du puzzle n'est pas forcément faisable unique-

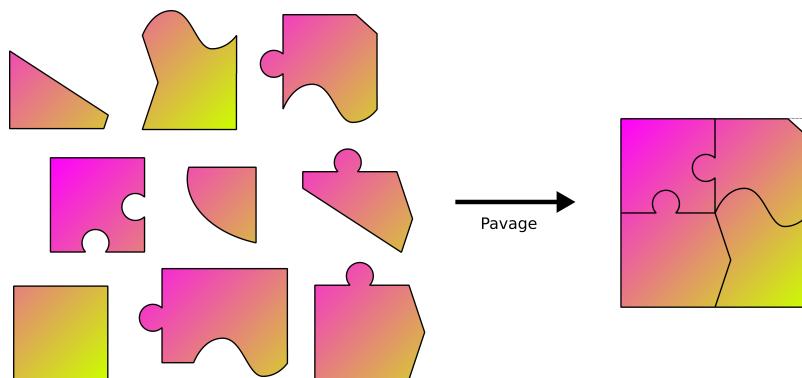


Figure 2.25: Phase de pavage. Choix parmi toutes les pièces disponibles pour former le puzzle le plus grand possible

ment à partir de ces pièces. De plus, parmi les pièces dont nous disposons, certaines se chevauchent. La phase de pavage consiste à choisir parmi les pièces, celles que l'on souhaite garder, en maximisant la couverture de notre puzzle et sans avoir de pièces chevauchantes (figure 2.25).

Dans l'idéal, toutes les pièces nécessaires à la reconstruction du puzzle complet sont présentes dans l'ensemble des pièces trouvées (voir figure 2.26). Dans le cas contraire, on ne peut pas reconstruire complètement le peptide à partir des résidus trouvés. Dans ce cas, notre but sera alors de trouver un assemblage le plus complet possible. Si beaucoup de résidus ont été trouvés lors de la première phase, il est également possible que nous ne trouvions pas qu'une solution unique de reconstruction. Nous n'aurons alors aucun moyen purement algorithmique, de choisir la meilleure façon de découper ces peptides.

2.4.3.2 Preuve de NP-Complétude

En 2015, j'ai eu la chance de participer, avec mes amis Thomas Nachtergale et Alexandre Temperville, à la finale du concours d'algorithme “Google Hashcode”. La veille de l'épreuve finale, un petit problème test nous a été proposé. Le but était de découper une pizza (matrice), composée soit de cases de jambon (H dans la matrice), soit de cases de sauce (T dans la matrice), en parts dites “royales”, en minimisant le gâchis de pizza (morceaux de pizza exclus de toute part royale découpée) (voir schéma 2.27). Les parts royales sont définies par des rectangles contenant *au minimum* 3 cases de jambon et représentant une *surface maximale* de 12 cases.

Ce problème a été prouvé NP-Complet sur le forum stackexchange [17] suite à la question d'un participant. La preuve a été faite par réduction du problème “Monotone

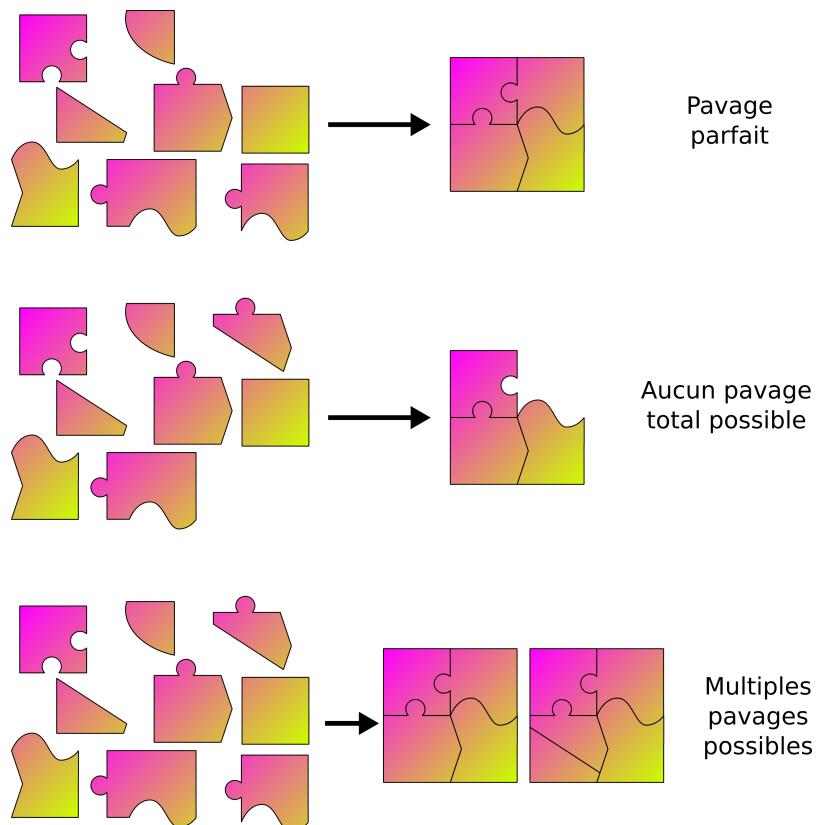


Figure 2.26: Phase de pavage. Choix parmi toutes les pièces disponibles pour former le puzzle le plus grand possible

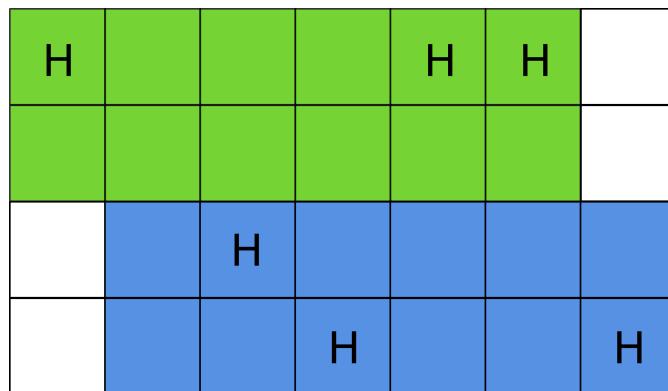


Figure 2.27: Exemple de pavage pour une pizza 7×3 . Ici la pizza a atteint un pavage maximal avec les deux parts verte et bleu (24 / 28).

cubic planar 1-3 SAT". Prouvons que le problème "Pavage" est également NP-Complet en utilisant une réduction du problème "Pizza". Définissons ce problème Pavage comme le problème de décision suivant : Connaissant tous les monomères candidats au placement

sur le peptide, est-il possible de trouver un sous ensemble de ces monomères qui soit non chevauchant et qui couvre 100% du peptide ?

Premièrement, il est très facile de montrer par certificat que pavage est NP. Le certificat contient l'ensemble des résidus pavés. Leur nombre est borné par le nombre de résidus issus de l'isomorphisme. Le certificat est donc de taille proportionnelle au problème. Deuxièmement, le validateur de certificat doit vérifier qu'aucun résidu ne chevauche un autre résidu. Pour cela, il est possible de vérifier les chevauchements pour toutes les paires de résidus du certificat. La vérification du certificat peut être faite de manière quadratique, ce qui permet de dire que le problème est NP.

Prouvons ensuite que nous pouvons résoudre le problème Pizza en utilisant le problème Pavage. Codons donc Pizza dans Pavage :

- La pizza de taille $n \times m$ sera représentée par un peptide rectangulaire de taille $n \times m$.
- Chaque case de la pizza sera représentée par un atome qui ne pourra prendre que les deux valeurs H et T selon le remplissage de la case.
- Le voisinage des cases sera représenté par des liens entre les atomes. L'atome correspondant à la case (i, j) sera lié aux atomes correspondants aux cases $(i - 1, j)$, $(i + 1, j)$, $(i, j - 1)$ et $(i, j + 1)$ (sauf sur les bords).
- Il est possible de générer en temps polynomial toutes les parts réalisables sur une pizza (car leur taille maximale est fixée). Chacune de ces parts sera représentée par un résidu placé sur le peptide comme si il avait été trouvé par l'étape de recherche de s2m. Par exemple une part allant de $(0, 0)$ à $(2, 3)$ sera représentée par un résidu couvrant tous les atomes entre ces coordonnées.

Avec ce codage, si nous arrivons à résoudre le problème de pavage, alors il devient alors possible de résoudre le problème de pizza.

Le problème de pavage est NP et il existe un problème NP-Complet qui peut se réduire en lui. Pavage est donc NP-Complet.

2.4.3.3 Pavage par optimisation linéaire

Nous allons ici résoudre le problème de pavage de manière exacte par l'utilisation de la programmation linéaire (LP). Un problème de programmation linéaire est un problème représenté par un ensemble de variables, une fonction de score ainsi qu'une liste de contraintes sur les variables. La fonction de score est accompagnée d'un objectif de maxi-

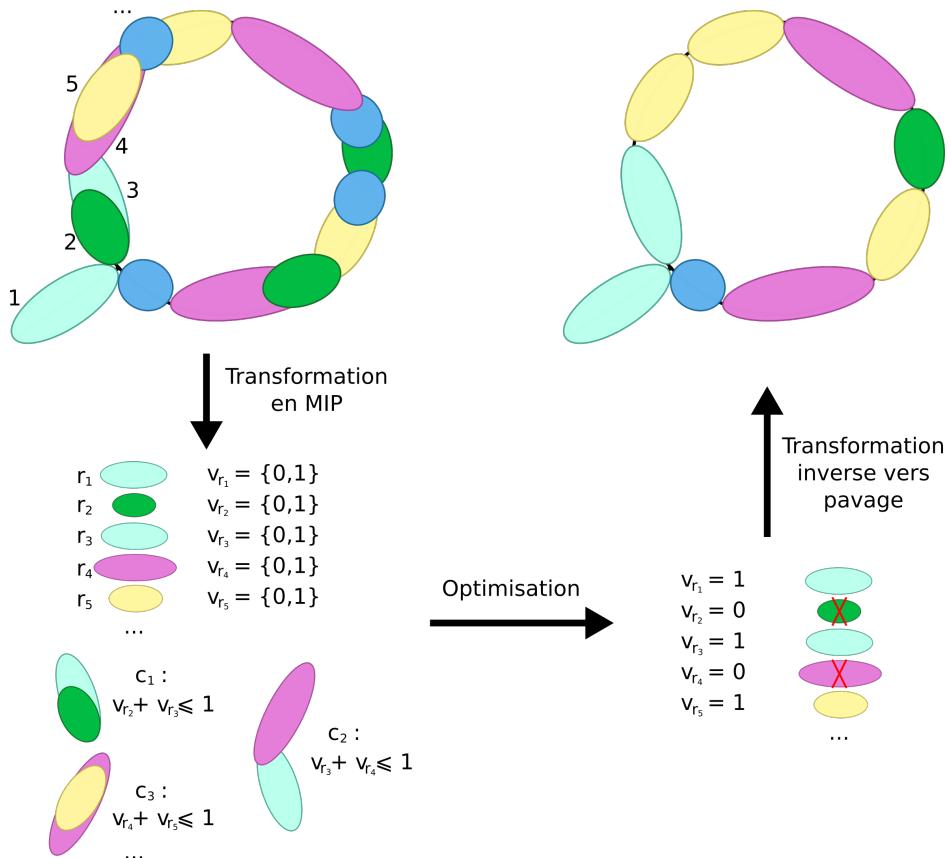


Figure 2.28: Représentation globale de la réduction du problème de pavage en MIP puis reconstruction de la solution

sation ou de minimisation. Ce type de problème est classiquement résolu par l'algorithme du simplexe [43]. Depuis la publication, de nombreuses variantes de l'algorithme ont été créées. L'une d'elle, appelée “Mixed Integer Programming” (MIP) [80], permet de résoudre des problèmes pour lesquels les variables sont des entiers.

Dans notre cas, nous allons rechercher les résidus qui seront utiles pour notre assemblage final. Ainsi, nous modélisons la présence ou l'absence dans la solution finale d'un résidu r_i trouvé durant la première phase par une variable binaire v_{r_i} (0 pour l'absence et 1 pour la présence).

$$\forall r_i, \quad \exists v_{r_i}, v_{r_i} \in \{0, 1\} \quad (2.14)$$

où r_i représente le i-ème résidu et v_{r_i} la variable binaire correspondant à r_i .

Nous souhaitons également interdire la possibilité de chevauchement de deux résidus. Nous pouvons caractériser cela par la restriction d'avoir au plus un résidu présent sur

chaque atome (*ie* une variable avec la valeur 1). On obtient donc une contrainte pour chaque atome ; ces contraintes étant représentées par le fait que la somme de toutes les variables des résidus présents sur un atome ne peut être supérieure à 1.

$$c_a : \sum_{r_i, a \in r_i} v_{r_i} \leq 1 \quad (2.15)$$

où a est l'atome correspondant à la contrainte c_a .

Enfin, nous cherchons à maximiser le nombre d'atomes couverts par les résidus présents. La fonction de score sera donc la somme des présences des résidus pondérée par la taille de chaque résidu :

$$\text{score} : \sum_{r_i}^{r_i} v_{r_i} \times \text{size}(r_i) \quad (2.16)$$

La réduction de notre problème de pavage en un MIP nous autorise l'utilisation des solveurs déjà existants pour trouver la solution exacte. Nous pouvons par exemple utiliser la bibliothèque GLPK (l'une des seules bibliothèques de LP sous licence libre).

Bien que cette solution soit élégante, MIP dans sa version ne contenant que des variables binaires fait partie des 21 problèmes NP-complets de Karp [30]. Il n'existe donc actuellement aucune méthode de résolution de ce problème en temps polynomial. Les cas les plus difficiles (*ie* contenant beaucoup de résidus chevauchants trouvés par isomorphisme) risquent de prendre beaucoup de temps. Proposons également une heuristique rapide pour résoudre le problème.

2.4.3.4 Pavage par heuristique gloutonne

L'un des moyens d'obtenir très rapidement une solution est de trier les résidus dans un ordre voulu, puis de parcourir la liste triée en ajoutant le résidu courant à la solution uniquement si il ne chevauche aucun autre résidu de la solution. On pourra, en théorie, toujours trouver au moins un arrangement de la liste qui atteigne la solution optimale. En faisant cela, nous reportons la difficulté du problème sur les critères de tri de la liste mais nous garantissons une très grande rapidité d'exécution. Le but ici est de s'approcher de ce tri optimal en

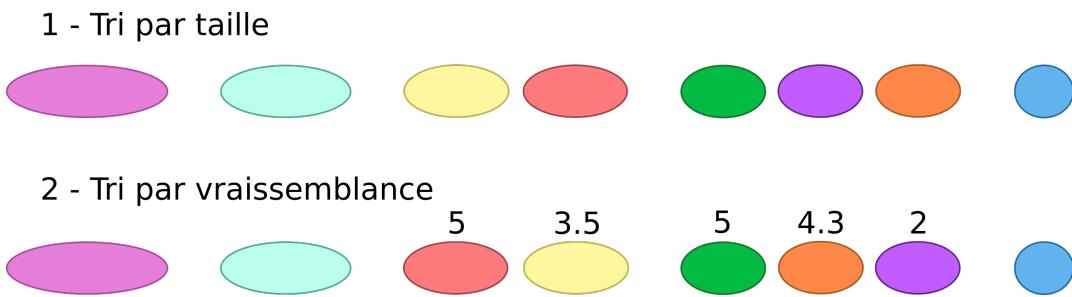


Figure 2.29: Tri pour le pavage glouton. Les résidus sont triés par taille puis par vraisemblance.

s’assurant d’une exécution très rapide de l’algorithme.

Données : Un peptide requête P et une liste de résidus RES pavés sur P lors de la phase d’isomorphisme

Résultat : Un sous-ensemble de RES constitué de pièces non chevauchantes sélectionnées par l’algorithme glouton.

Soit SOL un ensemble solution de résidus, initialement vide;

Trier RES selon les critères gloutons;

pour $r \in RES$ **faire**

si r ne chevauche aucun r' tq $r' \in SOL$ **alors**
 | Ajouter r à SOL ;
 | **fin**

fin

retourner SOL ;

Algorithme 1 : Algorithme de pavage glouton

Cet algorithme nécessite un tri sur une liste de n résidus candidats ($O(n\log n)$) puis consomme ensuite un élément de la liste à chaque tour de boucle ($O(n)$). Contrairement à la résolution exacte présentée précédemment, si la fonction de comparaison pour le tri ne dépend pas de n cet algorithme est linéaire. Nous pouvons donc garantir la rapidité d’exécution si les critères de tri sont également calculables en temps polynomial.

Le but est d’opter pour l’efficacité maximale locale en espérant que l’efficacité globale sera également maximale. Dans notre cas, la probabilité de trouver par hasard un résidu dans un peptide est inversement proportionnelle au nombre d’atomes qu’il contient. On peut donc dire qu’un résidu est d’autant plus “efficace” que sa probabilité d’apparition par hasard est faible. Le premier critère consiste donc à trier les résidus par taille décroissante (nombre d’atomes dont hydrogènes). C’est un critère classiquement utilisé pour la résolution gloutonne du problème du sac à dos [2].

Lorsque les résidus sont issus de monomères très similaires en taille, il arrive fréquemment que nombre d'entre eux aient exactement le même nombre d'atomes. Dans ce cas, le critère de taille n'est plus suffisant pour le tri et il nous faut en ajouter d'autres.

En plus de leur taille, les résidus peuvent être caractérisés par leurs types de liaisons chimiques. Chaque résidu forme des liens qui peuvent être différents d'un résidu à l'autre. Certains liens apparaissent fréquemment alors que d'autres moins, ce qui nous permet en théorie d'estimer les probabilités d'apparition d'un lien par rapport à un autre. Ainsi, on peut considérer qu'un résidu contenant des liens qui apparaissent le plus souvent a plus de chance d'être le bon résidu face à un second de même taille contenant des liens moins fréquents. Cependant, nous ne possédons pas de statistiques sur les fréquences d'apparition des types de lien.

Pour pouvoir tout-de-même classer les résidus de cette manière, nous requérons l'expertise de l'utilisateur. Avant la phase d'indexation, l'utilisateur doit définir les différentes règles de liaison qu'il veut voir apparaître dans les monomères. C'est à ce moment que nous lui demandons également de remplir une valeur pour chacune de ces règles. Cette valeur représente le poids que l'utilisateur souhaite donner à chacune des liaisons possibles. Ce poids représente la confiance que l'on donne à l'apparition d'un type de liaison. Ils nous permettent de calculer une confiance moyenne par liaison effectuée par le résidu. Plus une moyenne sera élevée, plus la confiance en le résidu sera forte et donc plus il sera présent dans les premiers résidus du tri.

Dans le cas où plusieurs résidus ne peuvent être départagés lors du tri, nous considérons qu'ils sont équivalents et leur ordre sera arbitraire.

2.4.3.5 Modulation de pavage

Le pavage glouton est un très bon moyen d'obtenir un résultat rapidement, mais il ne garantit pas sa qualité. Il est possible que certains "gros" monomères aient pris la place de plus petits et que le pavage soit dans une impasse. Dans ce cas, alors que la phase d'isomorphisme a identifié tous les monomères qui sont effectivement présents dans le peptide, le pavage correct n'est pas directement trouvé, l'algorithme nous laisse alors avec une solution partielle. Lorsque qu'un cas de pavage incomplet se présente, nous pouvons donc nous demander s'il ne serait pas possible d'obtenir la solution complète avec les isomorphismes déjà effectués. Nous proposons de repartir de la solution partielle et proposons d'effectuer une recherche locale pour tenter rapidement de faire mieux.

L'algorithme de modulation que nous allons vous présenter permet d'être certain de

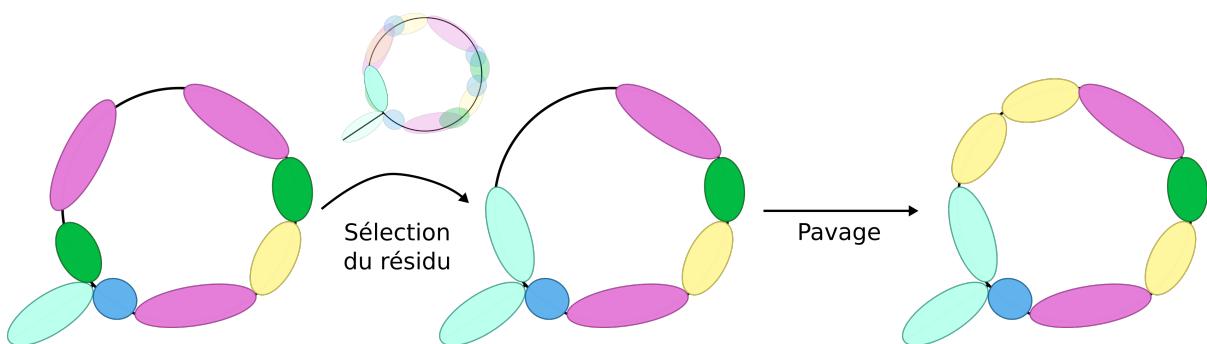


Figure 2.30: Représentation d'une étape de modulation. La première transition se fait par l'ajout forcé d'un résidu non choisi lors du pavage et de la suppression de tous les résidus le chevauchant. La seconde transition est effectuée par une étape de pavage glouton sur les zones non couvertes du peptide. Si une modulation ne suffit pas, cette étape est répétée de multiples fois au sein d'un algorithme de backtracking.

passer par la meilleure solution si elle existe. C'est un algorithme de backtracking classique qui passe par chacune des configurations possibles. À chaque étape, l'idée est d'utiliser un résidu qui permet de couvrir une zone libre du peptide (qui a donc été écarté par le pavage précédent) et de l'ajouter au pavage (voir figure 2.30). Utiliser ce résidu nous impose d'enlever de la solution les résidus chevauchants. En priorisant les résidus non choisis du début de la liste triée utilisée par le pavage (voir 2.4.3.4), nous essayons de trouver des solutions le plus rapidement possible. Si la solution n'est pas complète à la suite de cet échange, nous relançons un pavage sur les zones laissées libres. Après ce pavage, si la solution n'est toujours pas complète, nous relançons récursivement la modulation en interdisant de réutiliser les résidus que nous avons enlevé. Enfin, si l'appel récursif ne parvient pas à une solution totalement couvrante, nous faisons un retour en arrière en changeant le résidu que nous avions choisi.

L'algorithme permet d'explorer toutes les solutions possibles jusqu'à obtenir la meilleure. Le problème majeur est que si une solution parfaite n'existe pas, l'algorithme de modulation va parcourir tout l'arbre des solutions et l'exécution ne sera pas plus rapide que l'exécution d'une résolution de MIP. Rappelons que le pavage glouton effectue des choix que nous supposons pertinents. Il n'est donc normalement pas nécessaire de modifier la solution en profondeur pour obtenir l'idéal. Si la solution n'est pas trouvée rapidement, il y a fort à parier que la solution idéale n'existe pas. On peut donc limiter la profondeur de notre algorithme de modulation en limitant la profondeur de la récursion. Cette limite de profondeur diminue fortement le temps de calcul mais ne garantit plus de passer par l'optimal. Cependant, si une solution n'est pas trouvée en quelques étapes de modulation, il est fort probable

qu'aucune solution ne soit possible.

2.4.4 Recherche approximative (light)

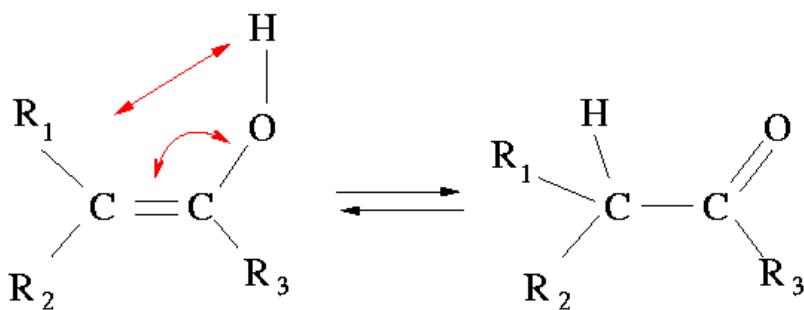


Figure 2.31: Tautométrie d'une molécule

Suite à des variations de composition ou de structure survenant dans certains résidus, quelques polymères ne sont pas découpés correctement en utilisant les algorithmes précédents. Il y a essentiellement deux cas pour lesquels l'isomorphisme ne fonctionne pas. Le premier cas concerne un *monomère ionisé* au sein du peptide, c'est-à-dire lorsqu'un proton a été perdu. Ainsi, lors de la phase d'isomorphisme, la fonction de comparaison d'étiquettes ne répond plus correctement car il manque un *H* sur l'un des atomes. Dans le second cas, c'est encore l'isomorphisme qui ne reconnaît pas un résidu car celui-ci est intégré sous un variant dit *tautomérique* (voir figure 2.31). Un tautomère de molécule est un isomère structurel de cette molécule. C'est-à-dire que la molécule tautomérique contient l'ensemble des atomes de la molécule initiale mais certains de ces atomes ne sont plus liés aux mêmes voisins. Dans le cas d'un tautomère, un noyau d'hydrogène change de position pour se lier à un atome voisin. Cette charge positive déplacée est compensée par une charge négative qui se déplace dans l'autre sens. Cette migration d'électron se concrétise par la relocalisation d'une double liaison. Ce changement peut même avoir lieu entre deux résidus différents d'un peptide.

Les deux transformations présentées ont pour point commun de ne pas modifier en profondeur la structure des molécules. Dans les deux cas seuls les "décorateurs" des étiquettes sont modifiés (hydrogènes et arité de liaison). Le moyen simple de reconnaître ce type de molécule est alors d'oublier ces décorateurs lors de la phase d'isomorphisme (voir figure 2.32). Dès lors, les résidus sont reconnus quelles que soient leurs variations sur les hydrogènes ou les liens. Bien que très séduisante cette méthode pose un problème de temps de calcul. La plupart des monomères NRP ne contiennent que des atomes C N O et H. Ne

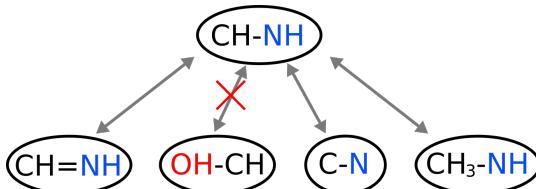


Figure 2.32: Exemple d’application de la fonction de matching *ligh* d’étiquettes. Puisque le nombre d’atomes d’hydrogène n’est plus un critère sélectif et que l’arité de la liaison chimique non plus, la comparaison est valide dès que les deux atomes non hydrogène sont identiques.

pouvant plus utiliser les H comme décorateurs et en rendant les liens indifférents les uns des autres, le nombre d’étiquettes différentes chute à 9. Il est donc compréhensible que le temps de calcul augmente en conséquence.

La solution est une nouvelle fois hybride. Dans un premier temps, nous cherchons rapidement une solution en effectuant un isomorphisme contenant toutes les étiquettes (isomorphisme *strict*), puis en effectuant un pavage. Dans un second temps si aucune solution complète n’est trouvée, nous relançons le processus en commençant par une nouvelle phase d’isomorphisme sans les décorateurs (isomorphisme *light*). Cette seconde phase d’isomorphisme est effectuée localement autour des zones qui ont posé problème lors de la première itération de l’algorithme. Cet algorithme en deux temps traite rapidement tous les cas simples avec des sous-algorithmes efficaces, pour revenir ensuite sur les cas les plus difficiles avec des algorithmes plus sensibles.

2.4.5 Vue globale des algorithmes

Tout au long cette partie, nous avons présenté par étapes les différentes parties de s2m. Nous rassemblons ici toutes les étapes au sein d’un schéma global (figure 2.33). La première partie du schéma (au-dessus des pointillés) est composée des deux phases de pré-calcul (voir 2.4.1.3). Cette étape de pré-calcul est gourmande en temps et en mémoire mais n’est à effectuer qu’une seule fois. Chacune des deux phases de cette étape se déroule en temps exponentiel mais reste calculable grâce à la petite taille des données. La partie principale de s2m possède deux alternatives algorithmiques. Dans les deux cas, l’algorithme commence par une phase de recherche d’isomorphismes (voir 2.4.1) afin de connaître les résidus candidats à la composition du polymère. Ensuite vient la phase de pavage, effectuée soit par une résolution de MIP (voir 2.4.3.3), soit par une heuristique de pavage (voir 2.4.3.4). Enfin, pour les cas plus difficiles, nous effectuons plusieurs passes de recherche

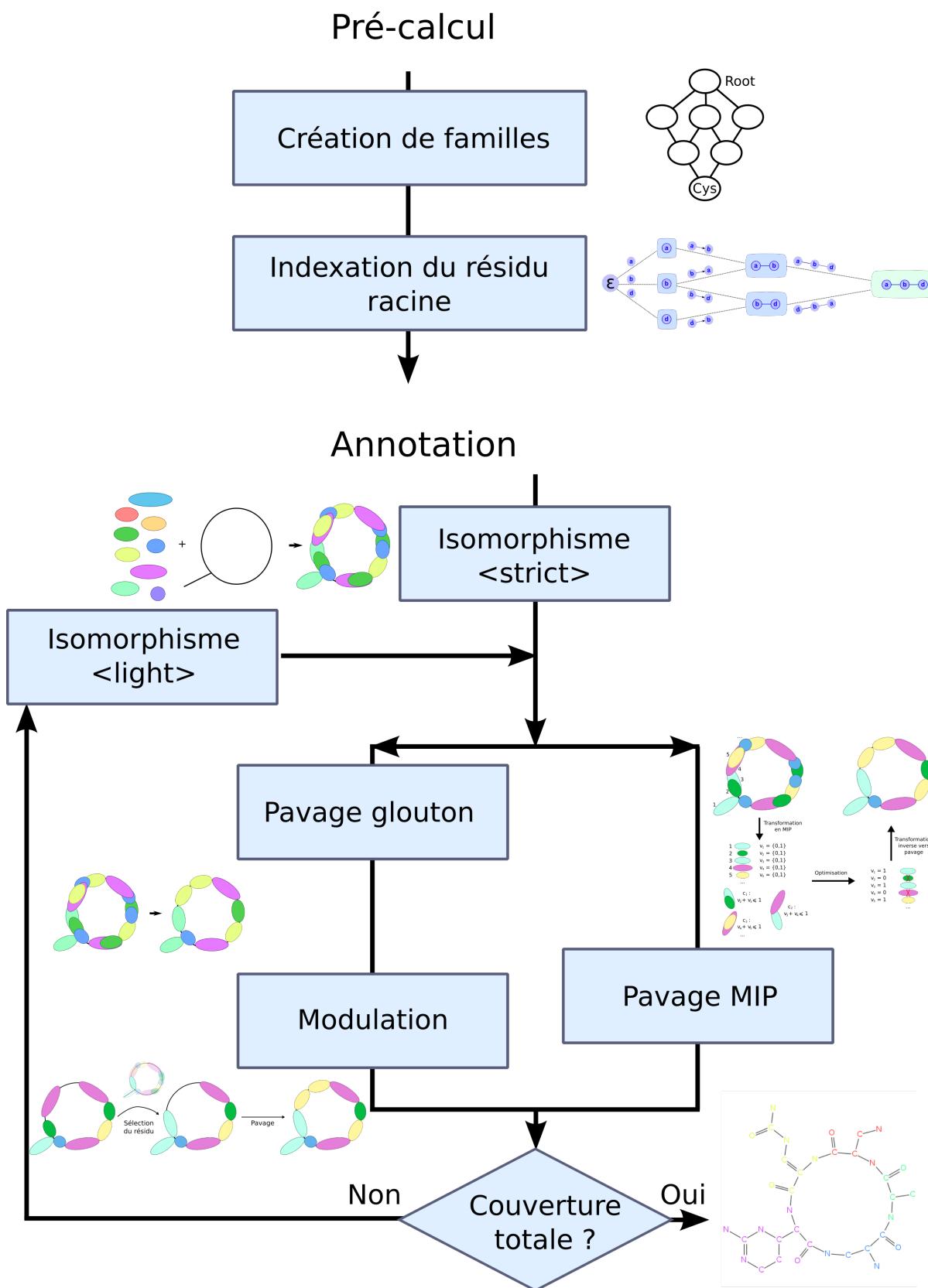


Figure 2.33: Schéma global de Smiles2Monomers

d’isomorphismes *light* (voir [2.4.4](#)) suivis par de nouvelles phases de pavage et modulation.

L’algorithme global est paramétrable. Pour le pré-calcul, nous pouvons fixer la variable *k* qui définira la taille de chaîne calculée avec le modèle markovien. Nous rappelons encore une fois que cette taille n’aura pas d’influence sur le résultat, mais uniquement sur le temps de calcul. Pour la partie recherche, nous devrons choisir le nombre de fois où nous effectuerons l’isomorphisme *light*. Ce paramètre est appelé “retry” (*r*) dans le logiciel (nombre de fois où l’on ré-exécute la boucle). Si nous optons pour la méthode heuristique de modulation, il est également nécessaire de définir une profondeur maximale de modulation (paramètre appelé “modulation” (*m*) dans le logiciel). L’influence de ces paramètres sur l’exécution de l’algorithme sera étudiée au sein de la section [2.5.3](#).

2.5 Résultats et interprétations

2.5.1 Jeux de données

2.5.1.1 Les constituants des jeux de données

Nous souhaitons connaître la pertinence des annotations issues de s2m. Pour cela, nous avons construit deux jeux de données de référence, au sein desquels les annotations biologiques attendues sont déjà connues. Ces bases de référence sont composées de polymères décrits à la fois par leur SMILES (représentation de leur structure atomique) ainsi que par l’annotation attendue (ici, un graphe monomérique). Sans la structure atomique, nous ne pouvons pas lancer s2m et sans structure monomérique, nous ne pouvons pas évaluer la qualité des résultats. Il existe très peu de bases de données contenant à la fois ces deux informations. Les bases Norine et CCD que nous allons présenter ici sont les seules, à notre connaissance, à posséder une grande quantité de polymères contenant ces deux structures.

Pour les entrées de s2m, nous constituons une base de monomères pour chacun des jeux de données. s2m pré-calculera les indexées pour chaque base de monomère puis s’exécutera sur chaque polymère. L’annotation de sortie sera alors comparée à l’annotation de référence pour en définir la qualité.

2.5.1.2 Premier jeu de données : Norine

Puisque le déclenchement de ce travail est dû à la volonté d’accroître le nombre d’annotations de NRP dans Norine, le premier jeu de données sera logiquement constitué de polymères non ribosomiques déjà annotés issus de cette base. Toutes les entrées de Norine

sont accompagnées d'annotations monomériques et c'est d'ailleurs ce qui fait la force de cette base. Nous avons donc extrait tous les NRP pour lesquels un SMILES était renseigné pour constituer un jeu de polymères de référence. La base de monomères est quant à elle constituée de l'ensemble des monomères connus et présents dans Norine.

2.5.1.3 Second jeu de données : The Chemical Component Dictionary (CCD)

CCD [77] est une sous partie de la base de données PDB [10] (Protein Data Bank). Cette base regroupe toutes les petites molécules présentes dans PDB. Parmi celles-ci se trouvent de nombreux polymères annotés ainsi que les monomères les constituant. L'avantage de cette base est que toute molécule présente est accompagnée de son SMILES. En extrayant les molécules annotées comme polymères, nous avons constitué un second jeu de données. Pour constituer la base de monomères dont s2m a besoin en pré-traitement, il nous a suffit d'extraire les molécules présentes dans les annotations polymériques.

2.5.1.4 Résumé sur les données

Les données de validation seront composées de deux jeux de référence distincts (figure 2.34). D'un côté les polymères extraits de Norine seront au nombre de 343. Ces données NRP seront accompagnées des 533 monomères de la base. Le second jeu est issu de la base CCD et comporte 378 polymères et 507 monomères issus des annotations. Il est à noter qu'aucun des polymères présents dans une base n'est également présent dans l'autre et que seuls 63 monomères sont communs aux deux bases. Ces monomères identiques sont en très grande majorité des acides aminés classiques et leurs variants proches.

2.5.2 Profil d'un résultat

s2m peut générer ses résultats sous différents formats. Il est possible par exemple de générer un format textuel *json* pour facilement recharger le résultat dans d'autres logiciels. Lorsque nous présenterons un résultat, nous le présenterons via le format graphique *html*.

Une page de résultat est composée de 3 parties (voit figure 2.35). En haut à gauche se trouve l'image du polymère étudié, colorée selon les monomères qui y sont présents. C'est l'information la plus immédiate car elle permet de visualiser immédiatement un résultat.

Sous l'image du peptide se trouvent trois listes de monomères. La première liste correspond aux monomères découverts dans la structure atomique par s2m et que l'on sait être corrects grâce aux annotations de référence. La seconde liste correspond également

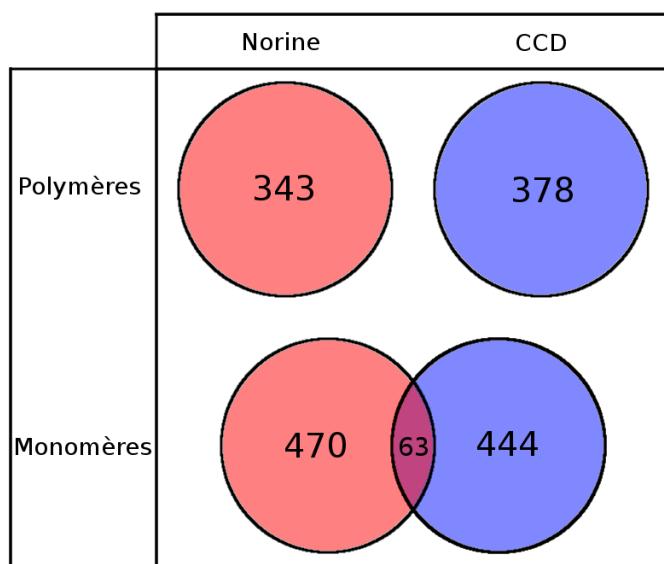


Figure 2.34: Répartition des données de validation

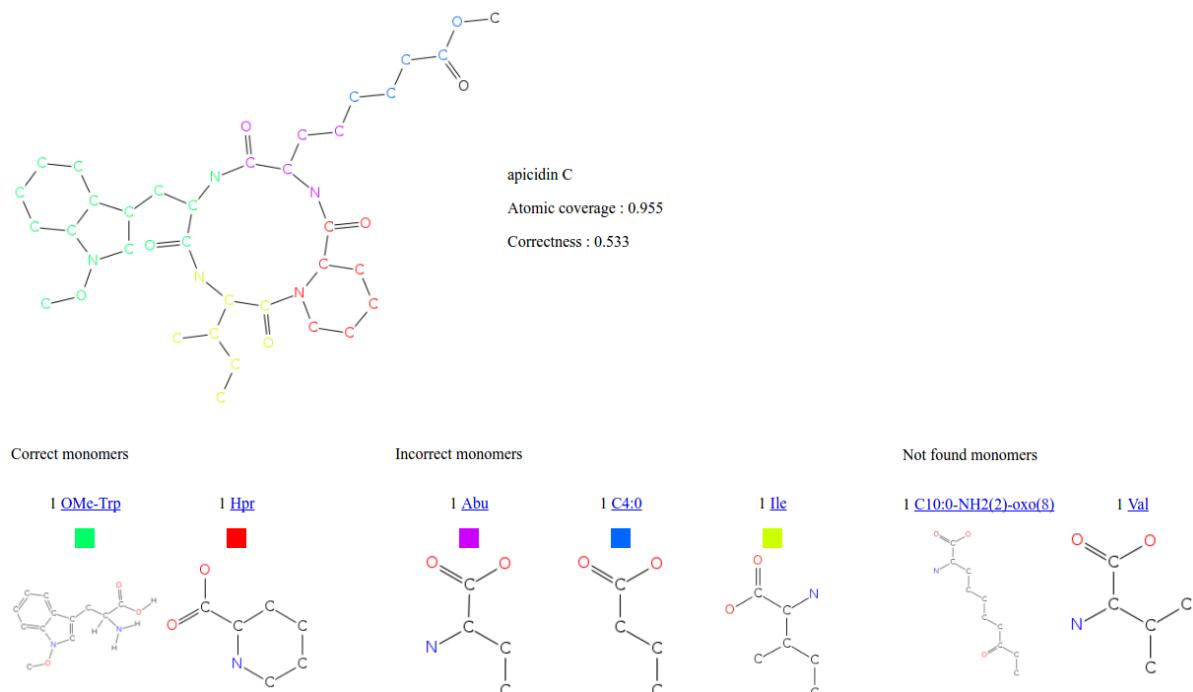


Figure 2.35: Le résultat HTML de l'exécution de s2m sur l'apicidine C

aux monomères découverts par s2m mais qui ne sont pas présents dans l'annotation de référence. La dernière correspond aux monomères qui sont présents dans l'annotation mais qui n'ont pas été trouvés par s2m.

La partie restante de la figure représente les informations générales sur le polymère. Tout d'abord, il y a le nom du polymère puis les statistiques de ce dernier. La première statistique est le taux de “couverture atomique” (*Coverage*).

$$\text{Coverage} = \frac{\text{nb atomes couverts}}{\text{nb atomes}} \quad (2.17)$$

Cette valeur donne le ratio entre le nombre d'atomes annotés par s2m comme faisant partie des monomères trouvés et le nombre d'atomes du polymère. La seconde valeur appelée *correctness*, donne également un ratio d'atomes, mais uniquement pour les atomes correctement annotés.

$$\text{Correctness} = \frac{\text{nb atomes couverts correctement}}{\text{nb atomes}} \quad (2.18)$$

Cette valeur ne prend en compte que les atomes faisant partie des monomères de la première des trois listes. Lors de la présentation des résultats, nous ferons toujours attention de bien mentionner si les valeurs sont des comptages monomériques ou des comptages atomiques. Les écarts entre ces deux valeurs peuvent être très importants à cause de la variabilité du nombre d'atomes composant les monomères.

2.5.3 Choix des paramètres

Nous allons tout d'abord nous poser la question du choix des bons paramètres pour effectuer nos tests. Nous reviendrons d'abord sur la façon de fixer le paramètre d'apprentissage pour le pré-calcul. Rappelons que, de ce paramètre, découlera le temps de calcul pour les peptides mais que la qualité des résultats ne variera pas. Puis nous étudierons la qualité des résultats obtenus en exécutant le programme pour plusieurs jeux de paramètres. Enfin nous analyserons le temps de calcul pour estimer les rapports entre temps et qualité et ainsi pouvoir recommander les meilleures calibrations de l'outil.

2.5.3.1 Choisir l'ordre k du pré-calcul Markovien

Pour rappel, l'isomorphisme de sous-graphe est effectué à partir de chaînes pré-calculées afin d'optimiser le temps de recherche (voir partie 2.4.1.5). Ces chaînes sont formées de deux sous-parties. La première sous-partie, de taille k, est optimale en temps de calcul selon une base d'apprentissage. La seconde sous-partie est générée de manière gloutonne.

Nous allons ici faire varier k et tester les temps de calcul sur les jeux de données. Pour

ne pas parasiter la mesure du temps, nous allons désactiver toutes les autres optimisations logicielles qui augmentent l'efficacité.

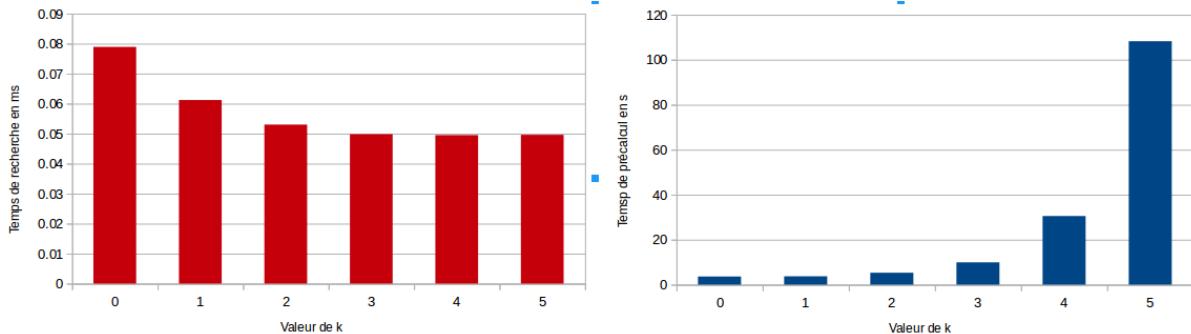


Figure 2.36: Évaluation de l'efficacité d'apprentissage des chaînes sur l'ensemble des monomères des deux jeux de données. À gauche : représentation du temps de recherche moyen d'une chaîne dans un peptide selon k. À droite : temps d'apprentissage pour tous les monomères selon k.

Observons les résultats sur le schéma 2.36. Le graphique de gauche représente le temps d'exécution moyen d'un isomorphisme (*i.e* recherche d'une chaîne dans un peptide) en fonction de k. Le graphique de droite représente le temps total d'apprentissage en fonction de k.

Quelles que soient les valeurs de k, la recherche est très rapide. On parle ici en centièmes de millisecondes. Lorsque la chaîne est entièrement générée de manière gloutonne, chaque isomorphisme prend environ 8 centièmes de ms. Ce temps descend à 5 centièmes à partir d'un k valant 3. Ce résultat est très intéressant car il nous montre qu'en moyenne seuls les 3 premiers nœuds sont utiles pour l'optimisation des chaînes sur nos jeux de données. Les monomères non détectables dans un peptide donné (même ceux constitués de plusieurs dizaines d'atomes), peuvent pour beaucoup être éliminés après 3 atomes si les chaînes sont correctement formées. Il n'est donc pas nécessaire de pré-calculer (sur nos données) des chaînes avec un k supérieur à 3. Ce pré-calcul fait gagner près de 35% de temps à chaque recherche de SI par rapport aux algorithmes de l'état de l'art.

2.5.3.2 Mesures

En général, lorsque l'on souhaite évaluer les résultats d'une prédiction, on utilise les statistiques découlant des vrais/faux positifs/négatifs. Commençons par définir chacune de ces catégories dans notre cas :

- Vrai positif (VP) : Un vrai positif est obtenu lorsque le résultat de la prédiction est

identique au résultat attendu. Dans notre cas, une annotation peptidique est un VP si tous les monomères présents dans l'annotation de s2m sont également présents au sein de l'annotation biologique et inversement. Nous aurons donc une couverture et un *correctness* de 100%.

- Faux positif (FP) : Un faux positif est obtenu lorsque le résultat de la prédiction paraît correct mais qu'il diffère du résultat attendu. Dans notre cas, une annotation est un FP si tous ses atomes sont couverts par l'annotation mais que celle-ci diffère de la référence. Nous aurons donc une couverture de 100% et un *correctness* inférieur à 100%
- Faux négatif (FN) : Un faux négatif est obtenu lorsqu'un résultat indique une erreur alors qu'il est possible d'obtenir un résultat correct. Dans notre cas, une annotation est un FN lorsque nous n'arrivons pas à produire une annotation complète du peptide. C'est à dire que nous avons à la fois une couverture et une *correctness* inférieures à 100%.
- Vrai négatif (VN) : Un vrai négatif est obtenu lorsque le résultat ne paraît pas correct et qu'il n'est pas possible d'en obtenir un. Dans notre cas, nous considérons les annotations de référence sont correctes : il est alors toujours possible d'obtenir une annotation couvrante puisque les molécules sont toutes issues d'un jeu d'annotations de référence.

2.5.3.3 Exécutions sur les jeux de données

Pour le choix des paramètres, nous n'allons, pour le moment, pas prendre en compte le temps d'exécution. Le but ici est d'exposer uniquement la qualité des résultats. Pour cela nous allons prendre comme indicateurs la **couverture atomique moyenne**, le ***correctness atomique moyen*** et les taux de VP/FP/FN. Nous allons faire varier le paramètre du nombre maximal d'exécutions de la branche *light* (R) ainsi que la profondeur de modulation pour le pavage avec modulation (M). Nous testerons également les deux algorithmes de pavage (MIP et heuristique).

Comme présenté au sein des tableaux 2.1 et 2.2, l'algorithme de pavage par optimisation linéaire n'est pas du tout sensible aux changements de paramètres. Ceci s'explique par le fait qu'il n'a besoin que d'une seule phase *light* pour parvenir à son optimal local. En effet, dans certains cas, le problème est tout-de-suite résolu après la première phase de pavage succédant à l'isomorphisme *light* car tous les monomères ont été correctement découverts. Dans les autres cas, l'algorithme du MIP effectue une sur-optimisation après

Params (R-M)	Heuristique				MIP			
	VP	FP	FN	Sensibilité	VP	FP	FN	Sensibilité
1 - 1	217	56	69	0.759	243	68	31	0.887
2 - 1	217	56	69	0.759	243	68	31	0.887
1 - 2	250	62	30	0.893	243	68	31	0.887
2 - 2	253	63	26	0.907	243	68	31	0.887
3 - 3	253	63	26	0.907	243	68	31	0.887

Table 2.1: Résultats de s2m sur les données de Norine. Dans la première colonne se trouvent les couples des valeurs des paramètres retry (R) et modulation (M)

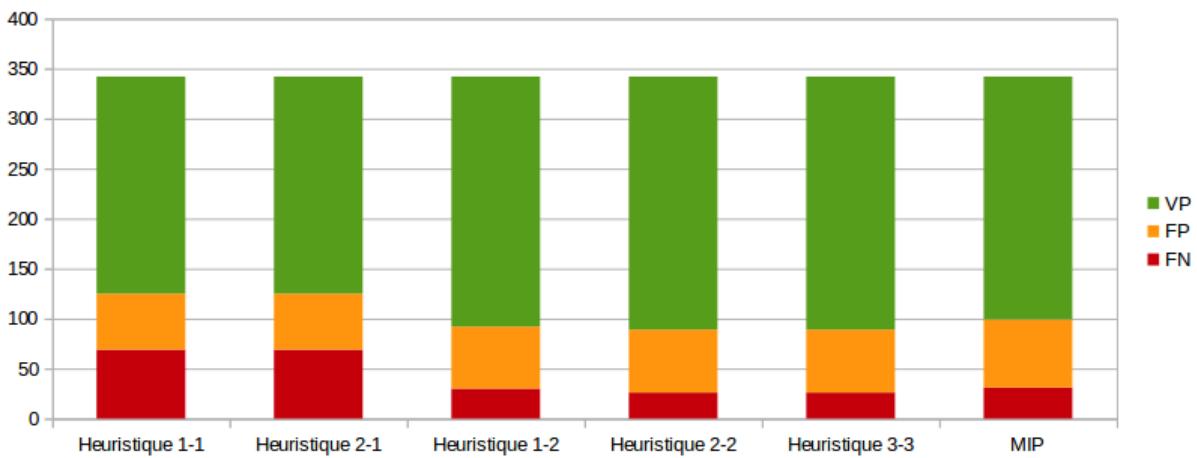


Figure 2.37: Diagramme en barre des résultats sur Norine

Params (R-M)	Heuristique				MIP			
	VP	FP	FN	Sensibilité	VP	FP	FN	Sensibilité
1 - 1	289	44	45	0.865	310	56	12	0.963
2 - 1	289	44	45	0.865	310	56	12	0.963
1 - 2	318	50	10	0.970	310	56	12	0.963
2 - 2	318	51	9	0.972	310	56	12	0.963
3 - 3	318	51	9	0.972	310	56	12	0.963

Table 2.2: Résultats de s2m sur les données de CCD. Dans la première colonne se trouvent les binômes des valeurs des paramètres retry (R) et modulation (M)

le premier pavage. Lorsqu'il n'est pas possible d'obtenir une couverture complète, l'algorithme a parfois recours à des placements de nombreux petits monomères. La solution est alors tellement éloignée de la réalité qu'elle devient inatteignable pour notre construction algorithmique.

Pour le pavage heuristique, la sensibilité aux paramètres est élevée. On peut voir que l'augmentation du nombre de modulations déclenche une augmentation du nombre de VP.

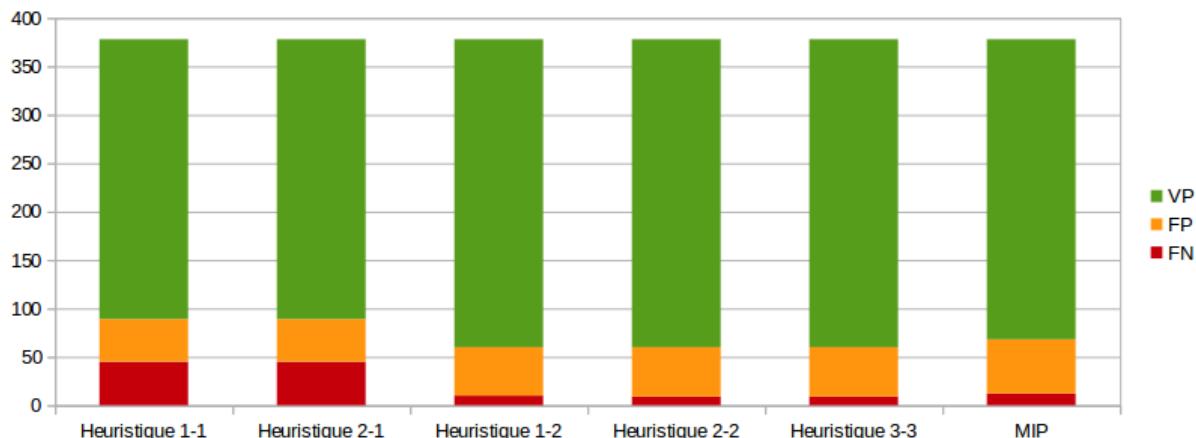


Figure 2.38: Diagramme en barre des résultats sur CCD

Le nombre de vrais positifs augmente significativement en passant le paramètre M de 1 à 2. Cette sensibilité s'explique par le fait que la solution optimale et la solution gloutonne ne sont pas immédiatement voisines. Il est souvent nécessaire de déplacer, non pas un, mais deux monomères pour atteindre l'optimal. Le paramètre R n'a que très peu d'influence et vient seulement compléter 3 résultats sur les données Norine et un seul sur les données de CCD.

En comparant les résultats des deux algorithmes, on s'aperçoit que l'algorithme comprenant l'heuristique de pavage obtient de meilleurs résultats (pour une modulation ≥ 2) que l'algorithme incluant un pavage exact. De nouveau, ce résultat s'explique par le fait que le pavage exact va trop loin dans l'optimisation. Si aucune solution totalement couvrante n'est possible avec les résidus issus de l'isomorphisme *strict*, alors le pavage maximal exact ira parfois créer des solutions globalement optimisées mais éloignées de la réalité. Dans ce cas, la recherche de résidus par isomorphisme *light* sert peu car les trous dans l'annotation ont été déplacés par l'optimisation. Il n'est souvent plus possible de construire l'annotation réelle, à moins de déplacer de nombreux résidus.

Au final, les résultats obtenus sont globalement meilleurs en utilisant l'heuristique gloutonne de pavage avec une modulation de profondeur M=2 et deux phases de recherche *light* avec R=2. Nous ne nous attendions pas à ces résultats et la sur-optimisation du pavage par MIP nous a surpris. Il est tout-de-même possible d'obtenir des résultats au moins aussi bons qu'avec l'heuristique, en effectuant uniquement une phase de recherche *light* à la place de la recherche stricte. Cependant le temps de calcul se voit multiplié par 10 sur nos données.

2.5.3.4 Temps de calcul global

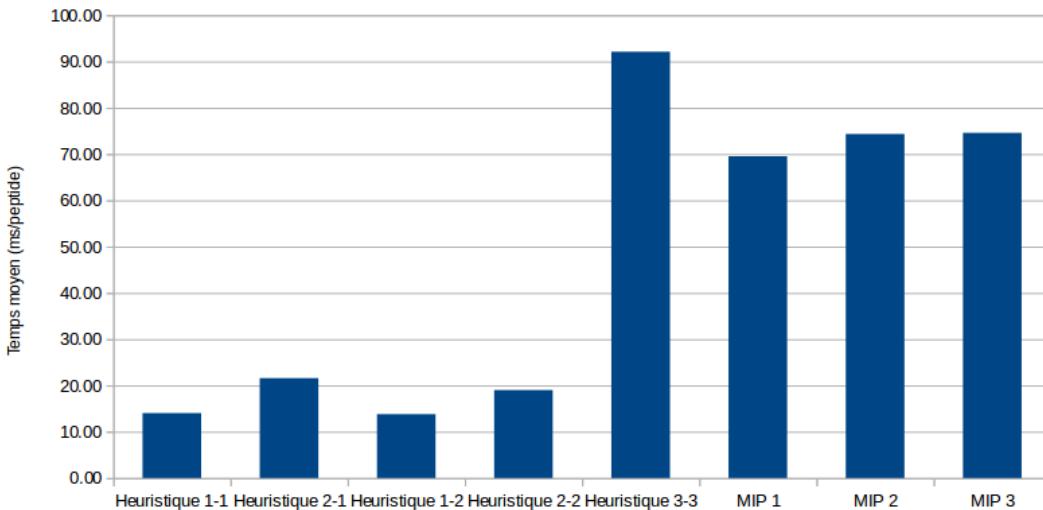


Figure 2.39: Temps d'exécution moyen par peptide (en ms) en utilisant les deux pavages et différents paramètres pour chacun d'eux.

Pour choisir un bon jeu de paramètres, il ne faut pas prendre que les résultats en considération, mais également le temps qu'il faut pour les générer. Pour cela, nous avons effectué une analyse comparative des algorithmes en utilisant les différents paramètres cités précédemment. Comme nous pouvons le remarquer sur le schéma 2.39, l'heuristique de pavage obtient un temps d'exécution beaucoup plus faible que le pavage MIP (excepté pour le pavage heuristique 3-3 qui n'a pas d'importance puisqu'il n'apporte pas d'amélioration de qualité). Ce résultat était attendu mais l'écart expérimental nous pousse fortement à utiliser l'heuristique de pavage plutôt que le MIP sur des données telles que celles présentées ici.

Comparons désormais les temps d'exécution pour l'heuristique en fonction des paramètres passés au programme. Le paramètre du nombre de boucles *light* (*R*) semble avoir le plus d'influence sur le temps d'exécution. C'est à nouveau un résultat que nous attendions puisque la recherche *light* ne s'appuie que sur une faible diversité d'étiquettes sur les noeuds (voir les sections 2.4.1.2 pour l'influence du nombre d'étiquettes et 2.4.4 pour la recherche *light*). Le temps passé à la recherche de motifs en utilisant les critères *light* est important. Répéter l'opération plusieurs fois multiplie logiquement le temps d'exécution.

Si l'on regarde bien les temps d'exécution, on peut également s'apercevoir que l'augmentation de la valeur de *M* permet de diminuer le temps d'exécution. Ceci peut s'expliquer par les cas où tous les résidus nécessaires à une couverture optimale ont été trouvés, mais qu'ils n'ont pas été pavés lors de la deuxième phase. Si la profondeur de modulation néces-

saire à l'inclusion de ces résidus n'est pas atteinte, alors le l'algorithme va passer son temps à effectuer de nombreuses recherches monomériques.

Pour conclure cette partie sur la sélection des paramètres, nous pouvons dire que nous recommandons l'utilisation de l'algorithme de pavage heuristique par rapport au pavage MIP. Cependant, il ne faut pas utiliser n'importe quels jeux de paramètres. Dans notre cas, pousser les paramètres de modulation ou de nombre de boucles à 3 ou au-delà est parfaitement inutile en terme de qualité de résultats et consommera énormément de temps. Utiliser un paramètre de modulation de 1 sera également une erreur puisque la qualité des résultats ne sera pas au rendez-vous. Il reste finalement deux alternatives intéressantes pour des données du même type que les notre. Soit nous privilégions la qualité à la vitesse et nous réglons les deux paramètres à la valeur 2, soit nous acceptons de perdre quelques bons résultats au profit de 30% d'économie de temps en réglant uniquement la modulation à 2.

2.5.4 Répartition des temps de calcul et analyse

Maintenant que nous avons choisi les paramètres, analysons attentivement chacune des étapes pour comprendre les points critiques et préparer de futures améliorations. Pour cela, nous avons exécuté le logiciel sur tous les peptides des jeux de référence en utilisant un pavage heuristique, une valeur de k à 3, une valeur de M à 2 et un R à 2. Pour chacun des peptides, nous avons stocké séparément les temps pour l'isomorphisme strict, le *light*, le pavage et la modulation. Les résultats sont affichés sur la figure 2.40. Les résultats sont triés par taille de peptide (nombre d'atomes).

Certains points extrêmes en temps de la figure ont été tronqués car ils écrasaient les résultats. 15 valeurs sur les 350 ont été redescendues à 50ms alors que certaines allaient jusqu'à plusieurs centaines de ms (le point max atteint 1s à lui seul). On distingue nettement deux comportements différents sur cette courbe. Soit les peptides ont été complètement analysés à la suite d'une seule passe de pavage (celle succédant la recherche stricte) ; dans ce cas, le temps d'exécution épouse une courbe qui augmente linéairement avec le nombre d'atomes ; soit le matching *light* et la modulation qui suit entrent en action et le temps devient alors fortement dépendant de la topologie du peptide (tous les points disparates).

Pour ce qui est de la première catégorie de peptides, on peut voir que la quasi-intégralité du temps est dépensé sur la recherche d'isomorphismes de sous-graphe. Comme nous le supposions au début de ce projet, les isomorphismes sont des goulots d'étranglement du fait de la complexité des algorithmes de résolution. Pour quelques peptides, le temps est

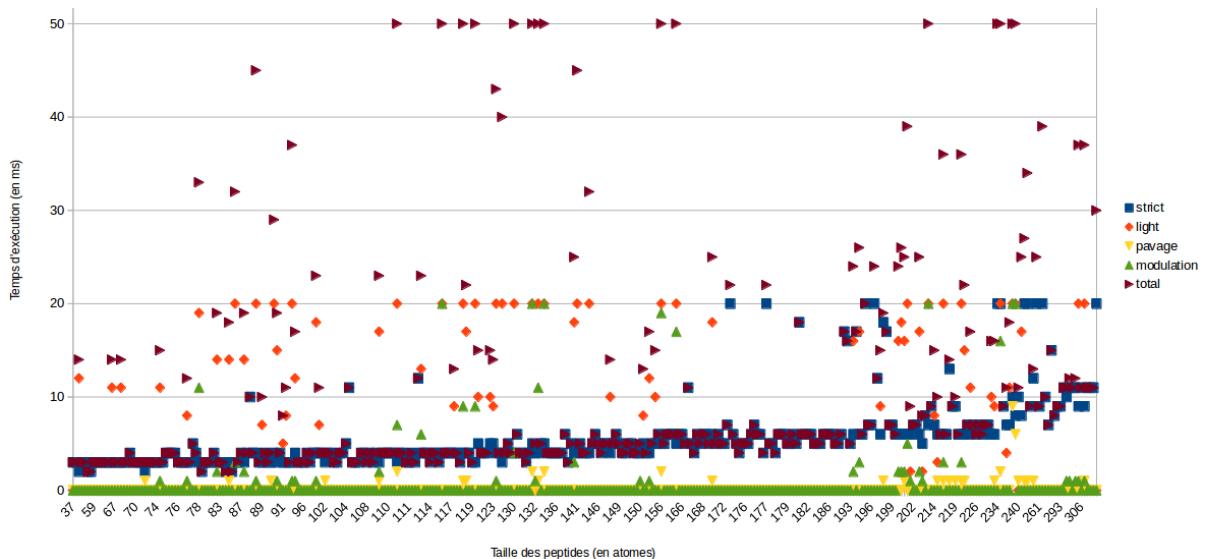


Figure 2.40: Temps d'exécution de s2m pour un extrait aléatoire de 350 molécules. Les données ont été tronquées au delà de 50ms pour ne pas écraser le graphique.

complété par une phase de modulation qui ne prend que quelques millisecondes. Dans le pire des cas, le temps est doublé par cette modulation. L'étape de pavage ne prend jamais plus de 3ms et n'est pas un problème pour le temps d'exécution.

Pour les points de la seconde catégorie, c'est à nouveau l'étape de recherche d'isomorphismes qui prend le plus de temps. Cependant, cette fois-ci, ce sont les étapes de recherche *light* qui prédominent le temps de calcul. Comme nous l'avions évoqué lors de la présentation de cette amélioration, le manque d'étiquettes disponibles diminue fortement l'efficacité de l'algorithme. En moyenne, un isomorphisme *light* est de 10 à 100 fois plus lent qu'un strict. De plus, les cas qui posent problème sont très souvent accompagnés de longues exécutions de modulation. Cependant, cet allongement du temps d'exécution permet de résoudre la majorité des cas. Sur l'extrait des peptides ici présents, seuls 20 d'entre eux finissent sur des impasses. Toutes les autres exécutions s'arrêtent en obtenant 100% de couverture.

Pour finir cette analyse des temps de calcul individuels, nous allons parler des cas extrêmes. Ces cas pathologiques masquent les réelles performances du logiciel. La moyenne de temps d'exécution globale est de 18ms par peptide mais lorsque nous retirons les 15 cas les plus critiques (ceux qui ont été tronqués sur la figure), cette moyenne chute à 10ms. À eux seuls, ces 15 peptides représentent plus de la moitié du temps d'exécution du logiciel. Le cas le plus critique (avoparcin-beta) représente à lui seul un cinquième du temps d'exécution total (plus de 1s). C'est une structure composée de beaucoup de cycles et d'em-

branchements. De plus, la structure atomique que nous possérons est fausse, ce qui rend impossible une couverture à 100%. L'algorithme passe énormément de temps en boucles *light* et en modulation. Nous pouvons en conclure qu'il faut éviter d'utiliser des boucles *light* sur des données dont les couvertures ne sont vraiment pas bonnes initialement. Si nous souhaitons par exemple analyser à l'aveugle un très grand nombre de molécules, il est préférable de se restreindre à une utilisation basique du logiciel, quitte à venir affiner les résultats intéressants par la suite.

2.5.5 Analyse des résultats

Pour analyser les résultats, nous avons exécuté s2m avec l'algorithme de pavage heuristique. La profondeur de modulation paramétrée à 2 et le nombre de passage par une phase *light* à 2 également. Vous trouverez tous les résultats sur les polymères testés à ces adresses : - CCD : www.TODO.com - Norine : www.TODO.com

Revenons sur les chiffres qui ont été présentés dans les tableaux 2.1 et 2.2. Un peu plus de 74% des annotations peptidiques issues de Norine et 85% de celles de CCD sont entièrement retrouvées par le logiciel. Mais il faut ici rappeler que les annotations ne sont plus considérées comme correctes dès lors qu'un seul monomère est faussement annoté. En regardant le taux d'atomes correctement attribués au sein des peptides, le taux de réussite augmente à plus de 93% pour les annotations de Norine et plus de 98% pour CCD.

Cependant, même ces très bons taux de réussite ne sont pas représentatifs de la qualité des annotations. En effet, en analysant chacun des résultats plus finement, nous nous sommes aperçu qu'il était possible de détecter des erreurs au sein des bases de données utilisées pour les tests. Analysons ensemble plusieurs cas afin de comprendre les raisons de ces erreurs.

2.5.5.1 Des peptides aux structures atomiques fausses

Beaucoup d'annotations fausses proviennent d'erreurs de structures atomiques pour les peptides. Prenons l'exemple de l'*enniatin I* (voir figure 2.41). L'annotation en sortie de s2m présente une couverture de 100% mais un *correctness* de 66%. En regardant les 3 listes de monomères, on peut s'apercevoir que l'annotation attendue comporte deux monomères nommés Hmp alors que le logiciel trouve deux 4Me-Hva. Les deux monomères se ressemblent : il est fort probable que, soit l'annotation monomérique, soit l'annotation atomique ait été mal saisie. Pour comprendre l'erreur, remontons aux sources des données. Sur la page Norine de la molécule, nous trouvons un lien vers Pubchem pour accéder à la

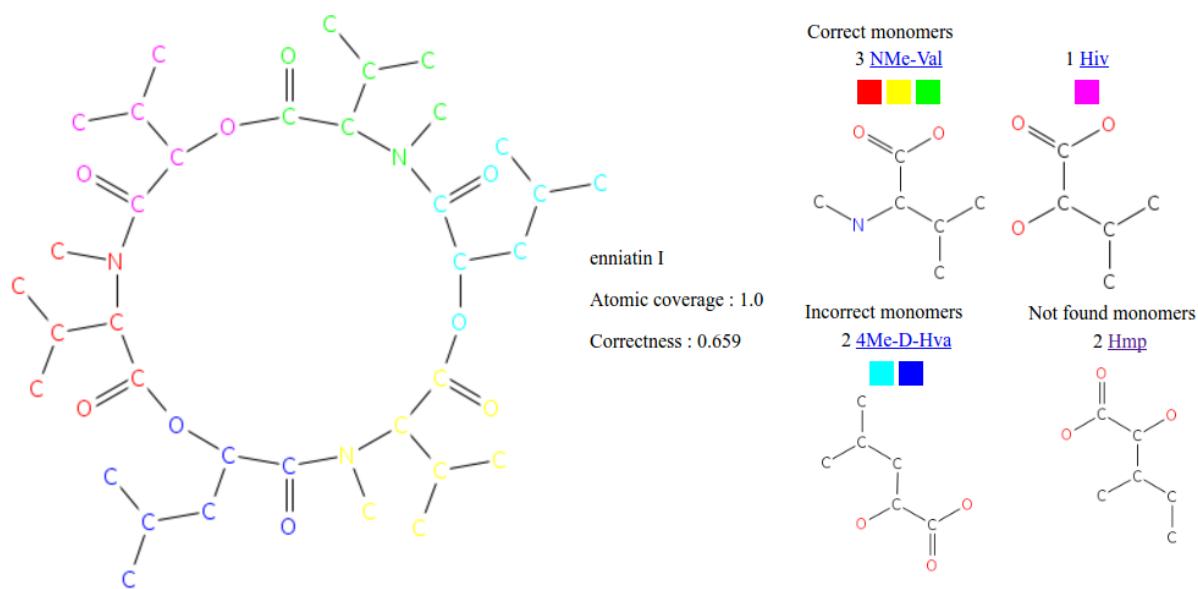


Figure 2.41: Résultat de s2m pour l’enniatin I. Deux monomères sont annotés faux ce qui nous mène à dire qu’une erreur s’est glissée soit dans le SMILES de la molécule soit dans l’annotation monomérique.

structure atomique et un lien vers la publication qui nous permet d’accéder à la structure monomérique.

En regardant la molécule présente sur PubChem (voir figure 2.42), nous pouvons constater que c’est effectivement la structure atomique qui a été utilisée pour le test. En consultant le contenu de la publication [62], on s’aperçoit que la structure monomérique correspond également bien à ce qui est présent sur Norine. Accompagnant cette annotation monomérique, un schéma présente la structure atomique. Cette seconde structure vient également confirmer la structure monomérique. Sans aucun doute, l’erreur provient de mauvaises données de PubChem entrées dans Norine. Le lien PubChem entré sur Norine n’est pas le bon et pointe vers un peptide similaire.

Nous n’avons pas fini de traiter manuellement toutes les données issues de s2m mais au moins 50% des erreurs semblent être de ce type.

2.5.5.2 Des monomères aux structures atomiques fausses

Lorsque le taux de couverture de l’annotation issue de s2m est inférieur à 100%, par définition, au moins un monomère n’a pu être trouvé. Il est possible que ce monomère n’ait jamais été répertorié, que la structure atomique du peptide soit incorrecte ou que la structure d’un monomère le soit également. Dans le cas de l’annotation de la dolastatin (voir

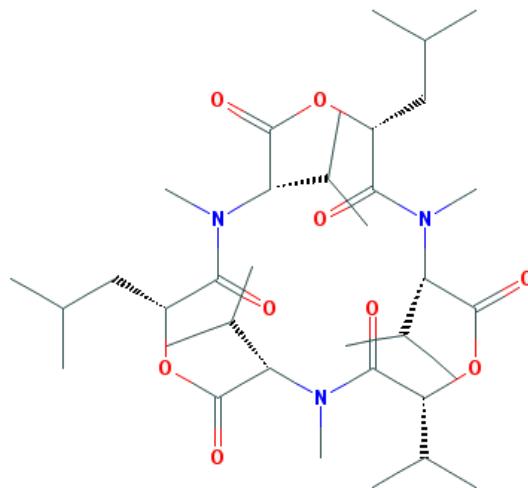
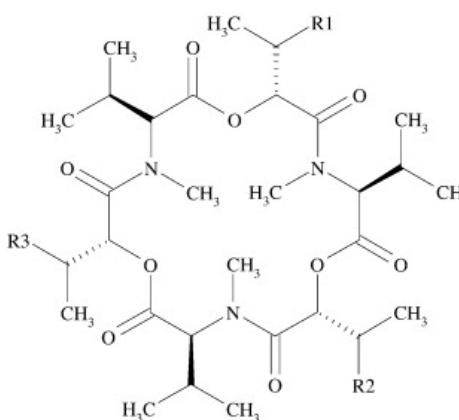


Fig. 1.

Chemical structure of the newly found enniatins.
For enniatin H, R1 is $-\text{CH}_2\text{CH}_3$, and R2 and R3 are $-\text{CH}_3$;
for enniatin I, R1 and R2 are $-\text{CH}_2\text{CH}_3$, and R3 is $-\text{CH}_3$;
for enniatin MK1688, R1, R2, and R3 are $-\text{CH}_2\text{CH}_3$.

Publication

Diversity in Beauvericin and Enniatins H, I, and MK1688 by Fusarium oxysporum isolated from potato

Pubchem

CID : 3010890

Figure 2.42: Structures atomiques publiées de l'enniatin I. Ces deux structures différentes nous aident à déceler une erreur dans la structure atomique de l'enniatin I sur Norine.

figure 2.43), c'est une erreur de structure atomique qui s'est glissée dans la base Norine. Le monomère Dap qui est sensé être présent n'est pas reconnu car l'atome d'azote n'est pas placé au même endroit au sein du monomère seul et du monomère dans le peptide. En cherchant sur Norine on peut s'apercevoir qu'aucun autre peptide de la base ne comporte ce monomère, ce qui veut dire qu'il a été ajouté spécifiquement pour ce peptide. Donc, le changement de structure du monomère n'est a priori pas dû à une modification de la molécule, mais ressemblerait plutôt à une erreur d'annotation. En parcourant l'article qui publia le peptide [?] et la structure atomique présente sur PubChem, on s'aperçoit que c'est effectivement une erreur de structure atomique uniquement présente sur Norine.

2.5.5.3 Erreurs d'annotations

Il arrive parfois que des molécules soient trompeuses. En entrant manuellement les annotations, il est possible de confondre un monomère avec un de ses dérivés proches. C'est ce qui est arrivé avec la fengycine A (voir figure 2.44). L'annotation effectuée par s2m détecte un monomère Gln alors que l'annotation présente dans Norine détecte un Glu. En effectuant à nouveau toutes les opérations de vérification de la structure atomique et monomérique, on s'aperçoit que l'article [?] donne raison à s2m. Cette fois-ci, c'est l'anno-

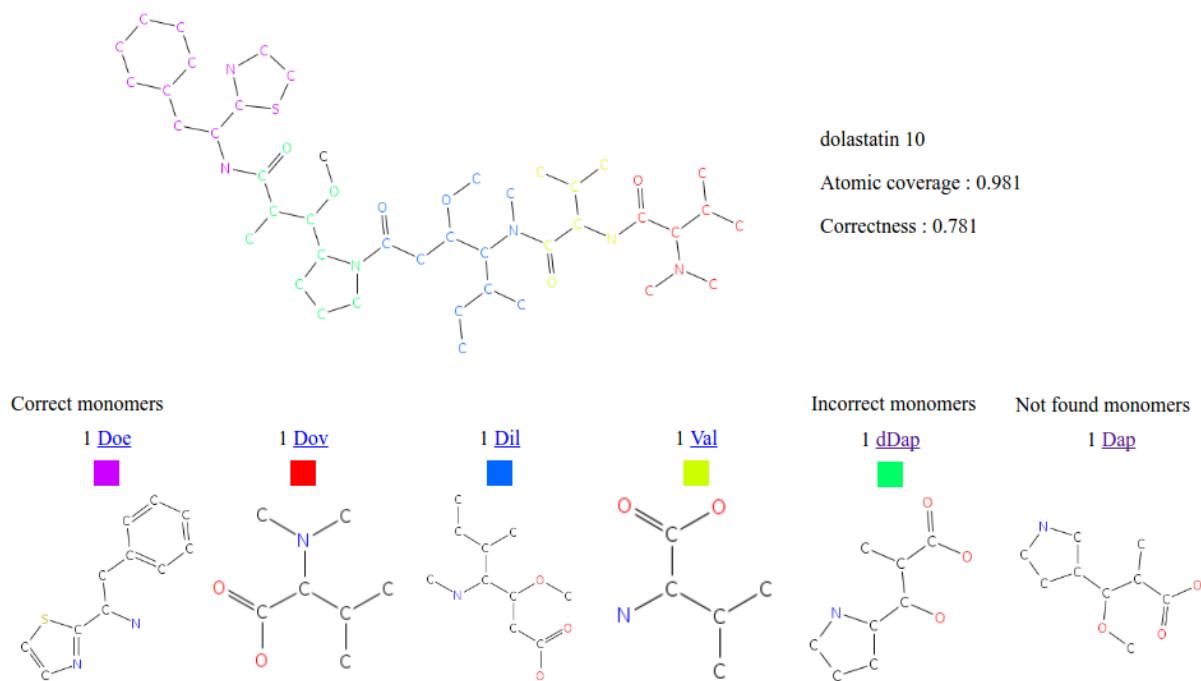


Figure 2.43: Résultat de s2m pour la Dolastatin (peptide de Norine). Une erreur de SMILES de monomère empêche une couverture totale du peptide.

tation monomérique présente dans Norine qui est à changer pour obtenir une information correcte.

2.5.5.4 Les équivalences

Le dernier type d'erreurs d'annotations commises par s2m est strictement dû aux algorithmes utilisés pour le pavage. C'est un problème d'annotations équivalentes. Comme nous le voyons sur le schéma 2.45, la "cyclotheonamide E" est un peptide couvert à 100%. De plus, l'annotation présente dans Norine propose deux monomères visuellement très proches des monomères suggérés par s2m (Arg/k-Arg et NFo-Ile/Ile). En regardant dans le détail, on peut observer que c'est un problème dû à une formylation entre deux monomères. L'un des deux monomères a été formylé, ce qui fait qu'un groupement C=O est présent entre les monomères Arg et Ile du peptide. Cependant, il pourrait *a priori* s'agir de n'importe lequel des variants. Dans les deux cas, l'annotation qui en résulte reste couvrante à 100%. N'ayant aucun moyen algorithmique de faire la différence, ici s2m a choisi la première solution trouvée. Pour connaître l'annotation réelle, il faut remonter à la production du peptide et comprendre les mécanismes de modification de monomères qui entrent ici en jeu. Cette erreur est présente moins d'une dizaine de fois dans les données déjà vérifiées et ne pourra

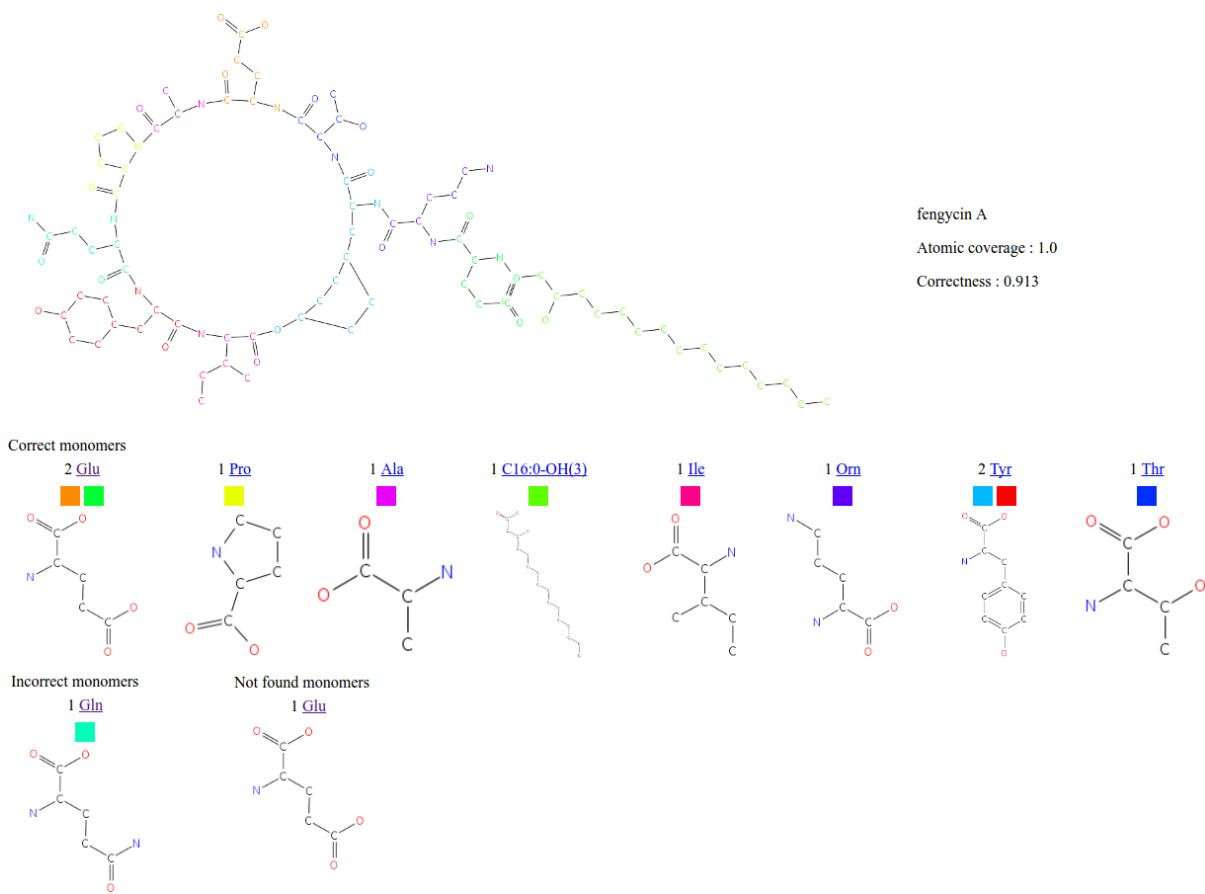


Figure 2.44: Résultat de s2m pour la Fengycin (peptide de Norine). Une erreur d'annotation entrée dans Norine réduit la *correctness* alors que s2m trouve la bonne solution.

hélas pas être corrigée sans prendre en compte plus d'informations biologiques sur le recrutement des monomères formant ce peptide non ribosomique.

2.5.5.5 Les analyses d'erreurs pour CCD

Durant toutes ces analyses, je me suis uniquement attardé sur les erreurs issues des peptides de Norine. Dans ces cas, il est possible de vérifier toutes les informations de cette base car de nombreux liens vers les origines des données qui y sont présents. Pour les données issues de CCD, nous détectons le même type de problème que dans Norine mais il est plus difficile d'analyser chacun d'eux en profondeur. Nous avons accès aux structures, mais plus difficilement aux publications associées. Nous pouvons donc également détecter la présence d'erreurs, mais sans forcément pouvoir expliquer leur raison. Seules les erreurs dues aux équivalences sont facilement détectables (voir par exemple 2.46). Pour les autres types d'erreurs, nous ne pouvons que spéculer sur leur origine.

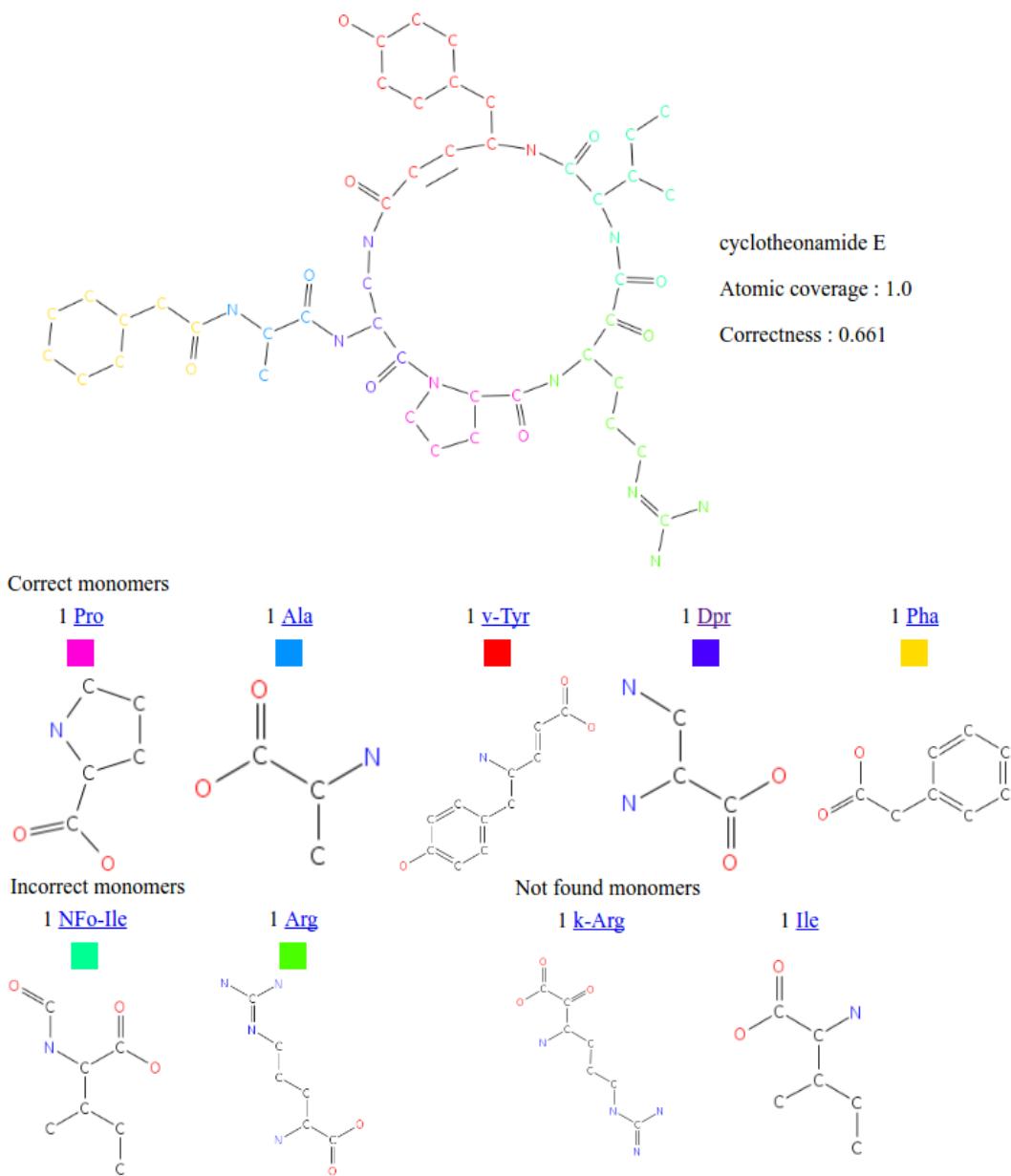


Figure 2.45: Résultat de s2m pour la Cyclotheonamide E (peptide de Norine). L'annotation monomérique recouvre l'entièreté des atomes du peptide mais n'est pas correcte à cause d'une équivalence entre monomères.

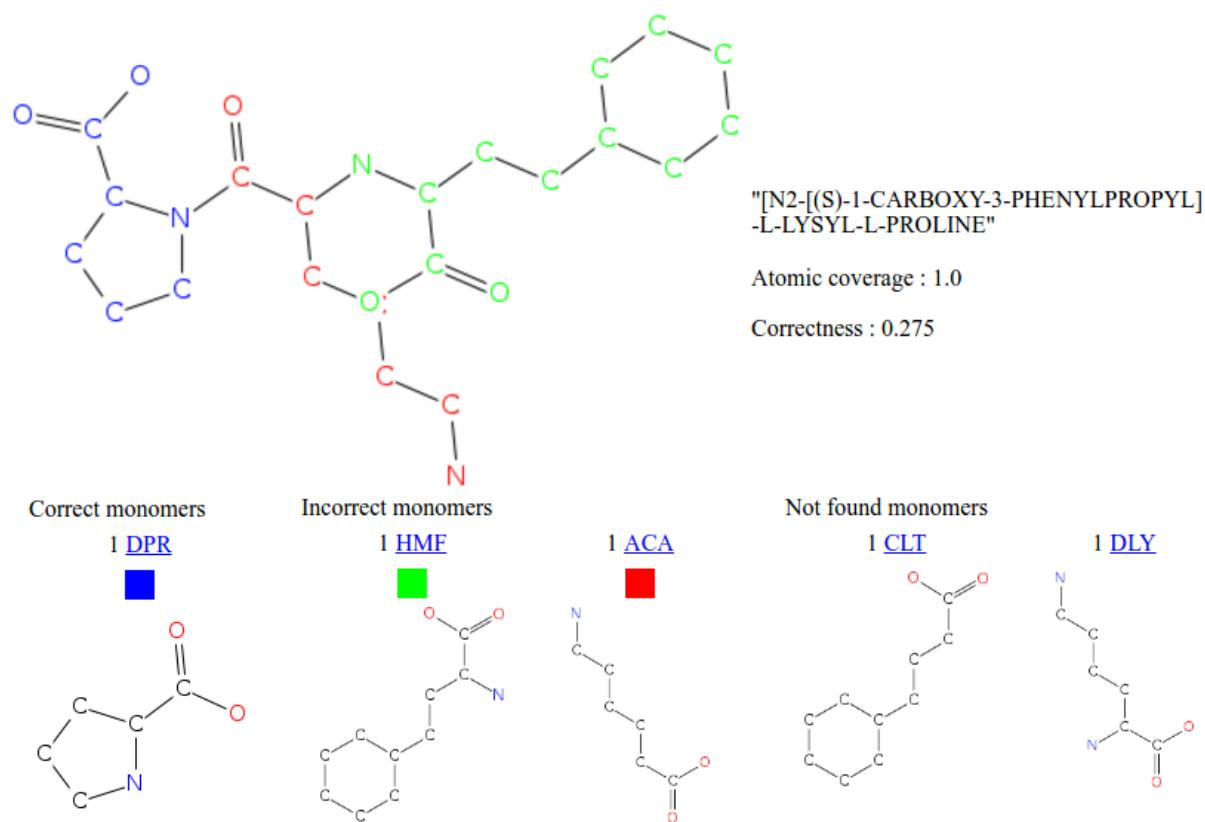


Figure 2.46: Équivalence entre monomères. L'atome d'azote inclus dans le HME pourrait également être inclus par le DLY mais nous n'avons aucun moyen de faire la différence.

Chapter 3

Vers un enrichissement de Norine

3.1 Norine

3.1.1 Généralités

Nous avons déjà plusieurs fois abordé la base de données Norine au sein de ce manuscrit mais nous allons ici décrire en détail son fonctionnement. Norine a été créée durant la thèse de Ségolène Caboche (2006-2009) dans le but de centraliser l'information sur les NRP. C'est la première base de ce genre à avoir été mise en place et c'est aujourd'hui la base de référence. Les structures atomiques et parfois monomériques de NRP étaient publiées dans des articles scientifiques puis quelquefois mises en ligne au sein de bases généralistes. Il n'existe souvent aucune annotation monomérique renseignée en dehors de ces publications. Norine a été créée dans le but de centraliser et rendre accessibles les annotations NRP connues. Sans cette centralisation, il est très difficile de créer des modèles de prédiction autour des NRP. Les données de Norine ont par exemple été très utiles pour la création de modèles de prédiction d'activités NRP [4, 5].

Norine est une base de données PostgreSQL contenant plusieurs dizaines de tables, toutes articulées autour de la notion de peptide (voir la version très simplifiée en figure 3.1). La table centrale des peptides contient toutes les informations propres au peptide telles que le nom, la famille ou les structures atomiques et monomériques. Les informations d'activités, d'organismes producteurs ou le détail de chaque monomère sont quant à eux présents dans des tables annexes pour qu'elles puissent être partagées entre différents peptides. Par exemple, une table est dédiée aux activités. Ces activités sont ensuite mises en relation avec les peptides via une table "présente". Au sein de cette table d'association, il est possible de trouver plusieurs activités par peptide et plusieurs peptides par activité.

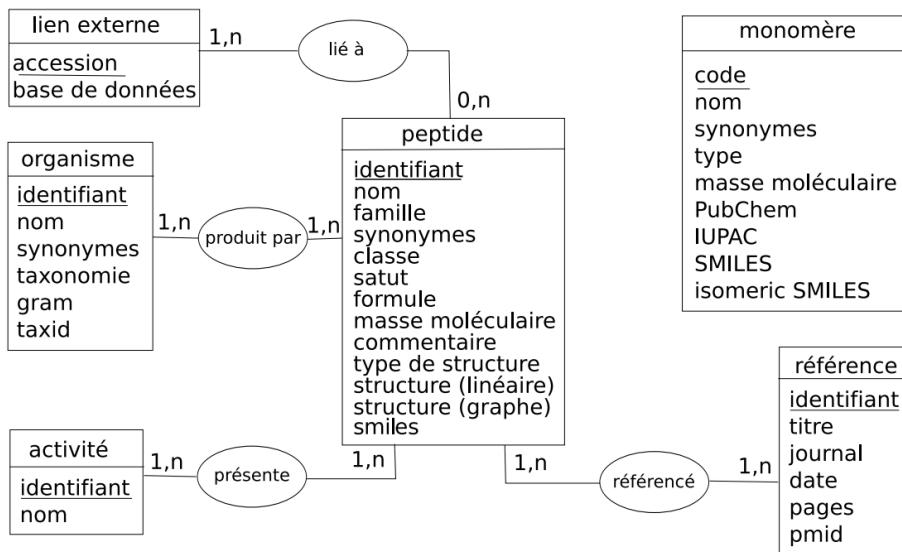


Figure 3.1: Schéma simplifié de la base de données Norine. Figure extraite de la thèse de Ségolène Caboche avec son accord.

La base de données est accompagnée d'une interface web et d'un serveur Java J2EE. L'interface a pour but de rendre les données simples d'accès pour tous les utilisateurs. Elle est accessible via le serveur de logiciels de l'équipe Bonsai à l'adresse <http://bioinfo.lifl.fr/norine/>. Chaque peptide et chaque monomère y possède sa page dédiée. Sur ces pages se trouvent tous les détails présents en base de données. Elles sont accessibles via des formulaires de recherche par nom/type/organismes producteurs/activité *etc.* Pour les peptides, il existe également une recherche par sous-graphe pour laquelle nous donnerons plus de détails par la suite.

3.1.2 Les peptides

3.1.2.1 L'interface web

Comme nous l'avons déjà évoqué, toute la base de données est axée autour de notre matière première : les peptides. Les pages centrales du site web sont donc logiquement les pages d'information correspondant à chaque NRP (voir figure 3.2). Ces pages sont divisées en trois parties. En haut de la page se trouvent les informations générales comme le nom, l'identifiant ou la liste des activités connues de ce peptide. La masse moléculaire est également présente et est particulièrement utile en spectrométrie de masse. Cette première partie est purement factuelle et ne fait que reprendre les éléments basiques potentiellement présents

malformin A1

Peptide

- Norine ID: NOR00627
- Activity: antitumor, toxin
- Class: peptide
- Formula: C₂₃H₃₉N₅O₅S₂
- Molecular weight: 529.71625 g/mol

Structure

- Type: other
- Number of monomers: 5
- Smiles: CCC(C)C1NC(=O)C(CC(C)C)NC(=O)C(NC(=O)C2CSSCC(NC1(=O))C(=O)N2)C(C)C
- Monomeric composition :

1
D-Cys

2
D-Cys

3
Val

4
D-Leu

5
Ile

Graph representation:

```

graph TD
    DLeu[D-Leu] --- Val[Val]
    Val --- D1[D-Cys]
    DLeu --- D2[D-Cys]
    D2 --- D3[D-Cys]
    D3 --- Ile[Ile]
    
```

Atomic structure:

Organisms

- Aspergillus niger
- taxonomy: cellular organisms, Eukaryota, Fungi, Ascomycota, Pezizomycotina, Eurotiomycetes, Eurotiales, Trichocomaceae, mitosporic Trichocomaceae, Aspergillus
- taxid: 5061

References

- Structure of malformin A2, reinvestigation of phytotoxic metabolites produced by *Aspergillus niger*
Sugawara F, Kim KW, Uzawa J, Yoshida S, Takahashi N, Curtis RW, *Tetrahedron Letters*, 1990, 31 (30), pp. 4337-4340
- Fungal malformins inhibit bleomycin-induced G2 checkpoint in Jurkat cells
Tomoda H, Omura S, Fukuda T, Hagimori K, Hasegawa Y, *Biological and pharmaceutical bulletin*, 2007, Aug,30(8):1379-83.
PubMed: 17666789

Links

[PubChem 4005](#)

Figure 3.2: Page de présentation de peptide dans l'interface Norine. L'image a été contracée pour passer en une page.

dans d'autres bases.

La seconde partie de la page représente la plus grande contribution de Norine. Cette

partie donne toutes les informations à propos des structures. Au minimum, elle contient la structure monomérique du NRP. Dans le meilleur des cas, la structure monomérique est accompagnée d'un SMILES et depuis peu, d'un dessin de la structure atomique dont chaque monomère a été coloré par s2m. La structure monomérique apparaît sous forme de graphe dans une interface visuelle ainsi que sous un format textuel. Chaque monomère entrant dans la composition est affiché dans une liste. Les liens au sein de cette liste permettent d'accéder au détail de chacun de ces monomères (nom, SMILES, composition, ...).

Enfin, la dernière partie de la page est consacrée aux références et liens. Ce sont toutes les informations qui permettront de replacer le peptide dans son contexte et d'en savoir plus au travers d'autres bases de données. Les organismes producteurs sont détaillés à cet endroit et un lien vers leur classification NCBI est disponible. C'est également ici que sont répertoriés les articles de recherche qui ont permis la découverte du peptide et l'établissement de son statut non ribosomique. Enfin, lorsqu'ils sont référencés, sont présents des liens vers d'autres bases de données. Actuellement, pour apprendre plus de détails sur les structures chimiques des molécules, sont présents des liens vers PubChem [?] et UniProt.

3.1.2.2 Les méthodes de recherche

Comme dans toutes les autres bases de données, il est possible de rechercher un peptide via de nombreux critères. En indiquant le nom ou l'identifiant, un peptide unique pourra être trouvé. En indiquant un nom partiel ou tout autre critère (avec la possibilité de combiner les critères), une liste de peptides sera retournée. Cependant, Norine propose également un autre type de recherche très intéressant pour notre type de données. Il est possible de rechercher les peptides par structures monomériques [?]. Plus précisément, il est possible de créer des expressions régulières de structures pour les rechercher dans la base. Prenons un exemple. Si nous souhaitons effectuer des recherches de peptides qui contiennent une Valine liée à n'importe quel variant d'une Cystéine ou d'une Leucine, il nous faut écrire la requête : Val,[Cys*|Leu*]@1@0. Ce format est un format de graphe. Les noms séparés par des virgules représentent les étiquettes des noeuds (un noeud Val et un noeud Cys* ou Leu*) et les valeurs séparées par des @, les arêtes. Pour résumer cette requête, nous avons :

- Le noeud 0 étiqueté par Val
- Le noeud 1 étiqueté par Cys* (n'importe quel dérivé de Cys) ou Leu*
- Un lien sortant du premier noeud (noeud 0) pour joindre le noeud 1 (@1 en première position)

- Le même lien sortant du deuxième nœud (nœud 1) pour joindre le nœud 0 (@0)

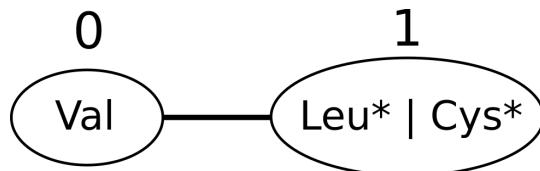


Figure 3.3: Requête Val,[Cys*|Leu*]@1@0.

Une fois la requête formatée, il est possible d'exécuter deux algorithmes différents. L'un des deux est un isomorphisme de sous-graphe. Nous avons déjà longuement abordé le problème d'isomorphisme durant le second chapitre (voir 2.3.2). La requête est croisée avec toutes les structures monomériques présentes en base et retourne tous les peptides dont la structure monomérique contient le motif recherché.

Le second algorithme ne prend pas en compte la structure mais uniquement la composition. C'est une recherche par *fingerprint* composée des comptages de monomères et des comptages par groupes de monomères [4]. Plusieurs groupements sont définis dans Norine. Par exemple Cys* cité précédemment est un groupement contenant la Cystéine, la D-Cystéine, l'alpha-méthyl-Cystéine et bien d'autres dérivés de la Cystéine. Les fingerprints des peptides en base sont pré-calculés et le fingerprint de la requête est créé à la volée. La requête fingerprint est comparée à tous les fingerprints de la base et une distance est calculée. Les résultats sont affichés par distances décroissantes. Cette recherche ne prend pas en compte les structures mais permet une souplesse dans les monomères inclus. Les regroupements permettent de trouver des peptides contenant des monomères proches mais pas identiques.

3.1.3 Les outils liés

Depuis la création de la base et du serveur web, beaucoup d'améliorations ont été effectuées [23]. Ici nous allons parler des trois dernières évolutions majeures.

3.1.3.1 Éditeur graphique

Mise à jour en 2015 par Juraj Michalik¹, l'interface de recherche dans la base possède désormais une interface graphique de création dynamique des requêtes. L'interface permet

¹Juraj Michalik était ingénieur d'étude pour la plateforme bilille au moment de la création de l'éditeur graphique.

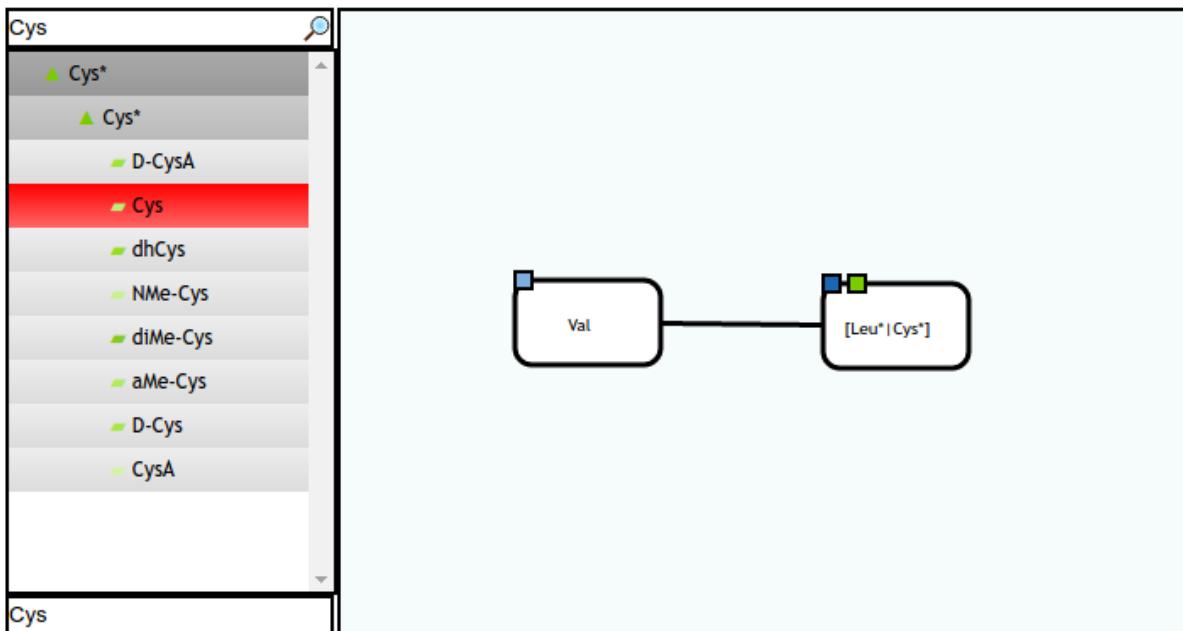


Figure 3.4: Éditeur graphique pour la construction de requêtes dans Norine. Ici, nous avons construit graphiquement la requête Val,[Cys*|Leu*]@1@0.

de générer facilement des requêtes complexes (voir figure 3.4). Le menu de gauche permet de sélectionner les différents monomères que nous souhaitons ajouter à une requête et la page principale permet d'ajouter des liens entre monomères par de simples clics. Au fur et à mesure de la création, la requête textuelle s'affiche au dessus de l'éditeur. Une version sans interface d'interaction de cet éditeur est également utilisée sur les pages des peptides de Norine afin d'avoir une représentation graphique de ceux-ci.

Cet éditeur est d'un réel intérêt pour la base car il permet une interaction utilisateur simple qui n'était pas possible auparavant. Une interface intuitive et conviviale est nécessaire pour rendre les outils bioinformatiques accessibles au plus grand nombre et correspond à une demande des biologistes utilisateurs.

3.1.3.2 Services REST

Initié en 2014 par Areski Flissi², une interface de récupération de données de la base Norine est disponible (interface REST). Cette interface a pour but la simplification de l'interfaçage des données Norine avec d'autres logiciels. Elle permet de récupérer un grand nombre

²Areski Flissi est ingénieur de recherche au CNRS. Il est pour le moment affecté à l'équipe Bonsai, équipe où j'effectue ma thèse, et s'occupe en grande partie de la maintenance et du développement de Norine et des outils satellites.

de données de Norine par des requêtes HTTP complexes, sans avoir besoin de passer par l'interface web. Nous avons utilisé cette interface pour récupérer les données de Norine afin de tester Smiles2Monomers. Une explication d'utilisation de l'interface est disponible à l'adresse <http://bioinfo.lifl.fr/norine/service.jsp>

3.1.3.3 Crowdsourcing via MyNorine

Le dernier outil que nous allons présenter s'appelle MyNorine. C'est un outil de crowdsourcing d'annotations NRP [23]. Ceci veut dire que toute personne, même extérieure à l'équipe de développement de Norine, peut facilement soumettre de nouvelles entrées. Tout comme l'interface REST, il a été développé par Areski Flissi.

MyNorine est en ligne depuis 2015 mais le développement de nouvelles fonctionnalités est toujours en cours. L'outil utilise une interface intuitive pour l'utilisateur. Il est possible d'entrer de nombreuses informations mais seules les informations importantes sont obligatoires. Parmi les informations obligatoires, on retrouve toutes les informations minimales comme le nom, le type de structure, la structure monomérique et au moins une référence bibliographique. Pour ne pas avoir à construire la chaîne de caractères compliquée représentant la structure, l'interface javascript décrite ci-dessus est disponible aux moments opportuns. Tout peptide soumis par un utilisateur sera ensuite envoyé à l'équipe de curateurs qui devra vérifier la validité et la pertinence de la soumission. Pour l'instant, cette équipe de curateurs est composée de Maude Pupin, Valérie Leclère, Areski Flissi et moi-même.

La mise à disposition de cette interface est une véritable plus-value pour la base de données. Elle permet de donner du poids à la communauté en lui confiant une part du travail d'archivage des NRP. À l'été 2015, nous avons organisé un workshop autour de la thématique NRPS/PKS durant lequel nous avons incité les membres de la communauté à partager leur travail au travers de MyNorine. Quelques peptides ont été soumis, mais il est encore nécessaire de communiquer pour rendre récurrent ce partage d'informations pour chaque nouveau peptide découvert.

3.2 Les contributions de s2m à Norine

Jusqu'à présent, nous avons présenté les méthodes informatiques qui permettent de traiter, analyser et générer les données contenues dans Norine. Cette section sera consacrée à la bioanalyse qui est nécessaire pour assurer la validation manuelle des données et donc leur

qualité. Comme tout au long de ma thèse, ce travail a été réalisé en étroite collaboration avec les membres du sous-groupe Norine. Les travaux que je vais présenter ici sont issus des analyses réalisées lors de nos réunions surnommées les “Norine days” et de collaborations avec des stagiaires et ingénieurs.

Les Norine days sont des réunions de travail qui réunissent tous les acteurs de Norine. Ces réunions se font lors de demi-journées et nous permettent d’aborder des problèmes et de proposer des solutions collectives. Elles m’ont été d’une grande aide pour acquérir les notions de biologie et de chimie nécessaires à mon travail. Lors de ces réunions, j’ai apporté les résultats de s2m, ce qui nous a mené à la découverte d’erreurs d’annotation présentes en base. Je présenterai par la suite les différentes étapes de la correction et de la mise en place de gardes-fous pour aider à faire disparaître les erreurs d’annotation dans les structures.

Durant ces journées nous prenons également beaucoup de temps pour analyser les nouvelles entrées soumises par des contributeurs extérieurs. Ces contributions nous ont d’ailleurs plusieurs fois menées à réaliser des changements au sein de l’ontologie de Norine et à la normalisation de concepts déjà présents.

La seconde partie de cette section sera consacrée aux collaborations avec des stagiaires et ingénieurs qui ont menées à la création de nouveaux contenus pour la base de données. Dans les deux cas que nous verrons, nous aborderons l’utilisation de s2m pour l’ajout automatique d’annotations.

3.2.1 Améliorations de l’existant

Norine est constituée d’environ 1200 peptides annotés. Toutes ces annotations ont été extraites une à une d’articles scientifiques puis entrées dans la base de manière manuelle. Bien que le travail ait été réalisé avec beaucoup de rigueur, un certain nombre d’erreurs se sont glissées en base. Comme présenté lors des résultats de s2m (voir [2.5.3.3](#)), le logiciel permet la détection de certaines erreurs. Voyons ici les procédures mises en place pour corriger ces erreurs et essayer d’empêcher l’introduction de nouvelles.

3.2.1.1 s2m pour corriger les erreurs

Lors de l’analyse des résultats nous avons détecté quatre types d’erreurs : les erreurs de structure atomique peptidique, d’annotation peptidique, de structure atomique monomérique et d’équivalences structurelles. Cette dernière erreur est entièrement due à la façon dont s2m est conçu et rien n’est à corriger en base. Pour les 3 autres erreurs, myNorine nous permet facilement de modifier les entrées de la base.

A : Modification via MyNorine

Dap

Logged as Yoann Dufresne

Add a new monomer to Norine

Code: Dap
Name: Dolaproline
Synonym: Dolaproline

Molecular formula: C9H17NO3
PubChem ID:
Molecular weight: 187.23618

Is clustered?

Parent cluster: Dap*

IUPAC:
Smiles: Cl1(NCC1)C(C(=O)=O)OC

PDB code:
Isomeric:

Search for peptides containing this monomer

Submit modifications

B : Couverture avant/après changement

Figure 3.5: Partie A : le processus de modification du monomère Dap au sein de Norine, via l'utilisation de MyNorine. Partie B : la conséquence de la modification du SMILES du Dap sur la couverture de l'annotation trouvée par s2m.

Appliquons une correction d'un peptide pour lequel nous avons découvert une erreur d'annotation. Pour la dolastatin 10, à la suite d'une recherche au sein des publications, nous avions déterminé que le monomère Dap de Norine était structurellement incorrect (voir 2.5.5.2). Effectuons la modification de ce monomère via myNorine (voir figure 3.5). Nous pouvons lancer la modification depuis la page Norine du monomère. Changeons le SMILES du monomère en plaçant correctement l'azote puis validons. Après validation de la modification, l'entrée est correcte en base.

Pour toutes les erreurs détectées par s2m, nous devons répéter ce processus de recherche des structures correctes puis de changement des entrées de Norine. Souvent, une modification dans un peptide nous amène à découvrir plusieurs autres modifications mineures nécessaires. Il nous est par exemple plusieurs fois arrivé de découvrir des incohérences de nomenclature et de finalement passer un long moment sur une procédure de normalisation. C'est un processus qui prend énormément de temps et qui doit être effectué collectivement afin de ne pas prendre de décisions contradictoires. Le traitement des erreurs est à ce jour

toujours en cours et nous avançons au rythme de 1 à 5 peptides par réunion (environ 100 cas d'erreurs à traiter).

3.2.1.2 s2m pour empêcher les erreurs

Après avoir traité les erreurs existantes, nous allons discuter de la manière de limiter l'introduction de nouvelles erreurs. Le seul moyen est d'essayer de détecter automatiquement les annotations fausses lorsqu'elles sont entrées et de proposer à l'utilisateur le choix entre notre correction et ce qu'il a entré.

Overview for Yoannin

Warning: Are you sure of the monomeric composition of the peptide? It seems than the monomeric composition you entered (Ala,Tyr,Arg,Ile,Glu,Ala,Adda) and the one infered from the smiles (bMe-Asp,Tyr,Arg,NMe-Dha,Glu,Ala,Adda) differ.

[peptide](#) | [structure](#) | [organism](#) | [references](#) | [links](#) | [comment](#)

Structure

- **Monomeric composition:** Ala,Tyr,Arg,Ile,Glu,Ala,Adda
- **Smiles:**
O=C(O)C2NC(=O)C(C)C(C=CC(=CC(C)C(OC)Cc1cccccc1)C)NC(=O)C(NC(=O)C(C)C(NC(=O)C(NC(=O)C(=O)C(=C)N(C(=O)CC2)C)Cc3ccc(O)cc3)C(=O)O)CCCN=C(N)N
- **Graph representation:** Ala,Tyr,Arg,Ile,Glu,Ala,Adda@1,2@0,5@0,6@5,4@3,6@1,3@2,4

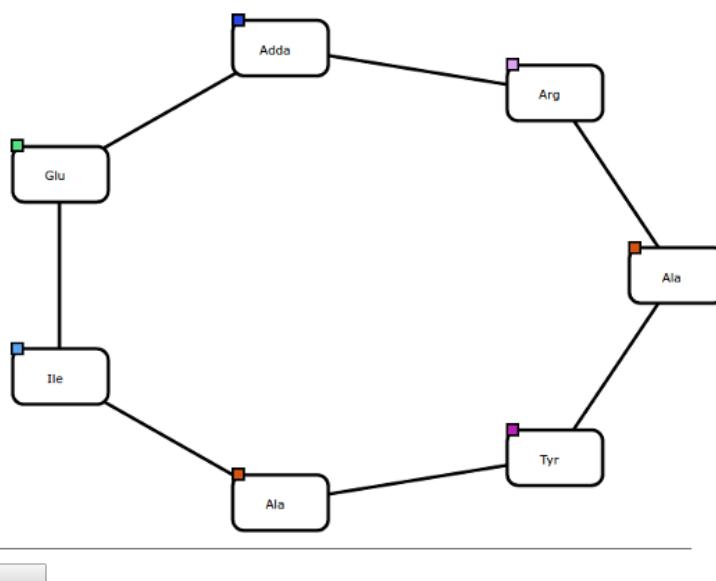


Figure 3.6: Alerte levée au sein de MyNorine lorsqu'une annotation d'utilisateur ne paraît pas correspondre à l'annotation automatique de s2m.

Dans le cas de Norine, toute nouvelle entrée manuelle est effectuée via MyNorine. Pour éviter l'introduction d'erreurs, si le SMILES de la molécule est entré, nous exécutons s2m en arrière plan. Si l'annotation obtenue par s2m couvre entièrement le peptide mais que cette annotation diffère de l'annotation entrée manuellement, nous avertissons l'utilisateur

et lui demandons de vérifier sa structure avant validation (voir figure 3.6). Cela nous a permis plusieurs fois durant les Norine days de nous rendre compte d'erreurs que nous étions en train de commettre et nous pensons que cela permettra d'en éviter de nombreuses autres.

3.2.1.3 s2m pour des saisies semi-automatiques

Figure 3.7: Création d'annotation à partir d'un SMILES lors de la création d'une nouvelle entrée via MyNorine.

Dans MyNorine, nous proposons à l'utilisateur d'exécuter s2m avant d'entrer son annotation de manière complètement manuelle. Si un SMILES est renseigné, alors s2m peut être déclenché pour obtenir une première annotation (voir figure 3.7). L'utilisateur peut ensuite la modifier si celle-ci n'est pas complètement exacte. Par ce système nous incitons les contributeurs à ajouter le SMILES de leurs peptides (information pas toujours présente dans Norine pour le moment) et nous évitons les erreurs d'inattention et fautes de frappe lors de la saisie utilisateur.

Comme le montrent les deux dernières sections, s2m a permis à MyNorine d'augmenter encore la qualité des annotations entrées en évitant des erreurs humaines.

3.2.2 Mises à jour de Norine

Les Norine days nous ont amené à travailler très régulièrement sur les données de Norine, ainsi que sur des données d'autres bases. Cette exploration en profondeur nous a révélé plusieurs faits. Premièrement, les données de Norine, comme celles de nombreuses bases, sont vieillissantes. Par exemple en suivant des liens PubChem, nous nous sommes aperçus qu'ils ne pointaient plus tous vers les bonnes structures. Nous avons également mis à jour le fait que l'évolution de la taxonomie au cours des années, a rendu obsolète certaines classifications d'organismes dans Norine. Plusieurs autres détails ont également changé et nous font dire qu'une mise à niveau s'impose. Cette mise à niveau sera le premier point que nous aborderons dans cette section.

Deuxièmement, en parcourant des bases telles que MiBIG qui répertorient des NRP et NRPS, nous avons constaté qu'il existe une grande quantité de NRP facilement accessibles et candidats pour un ajout au sein de Norine. Ces données sont en général bien structurées et pourraient facilement être indexées par Norine.

Enfin, il existe également de nombreuses molécules NRP présentes dans des bases de données généralistes. Nombre d'entre elles sont sans annotation monomérique. En utilisant s2m sur de très grandes bases telles que PubChem, nous essayerons d'obtenir des annotations monomériques afin d'extraire des candidats NRP. Ces molécules pourront ensuite être analysées manuellement, pour être enfin entrées en base ou ignorées.

3.2.2.1 Mise à jour des données de Norine

Nous allons ici parler du vieillissement des données de Norine. Il existe au sein de Norine des champs dont les informations sont obsolètes ou imprécises. Nous pouvons regrouper ces informations sous quatre catégories : les liens externes rompus (liens vers PubChem et UniProt), les taxonomies non mises à jour, les molécules/organismes ayant changé de nom et les mises à jour de peptides associés à de nouvelles publications. Analysons un à un chacun de ces cas.

Vérification des liens externes Les liens rompus représentent la catégorie la plus facile à traiter. Un script simple a suffi pour les corriger. Un à un, nous suivons automatiquement les liens vers les bases. Certains d'entre eux ne mènent plus à rien et nous pouvons alors les supprimer. Dans ce cas, nous cherchons ensuite le nom du peptide via l'API de

PubChem pour espérer trouver une autre page correspondante. Lorsqu'une structure est retrouvée, nous la faisons annoter par s2m afin de vérifier la cohérence de structure avec l'annotation manuelle de Norine. Si l'annotation correspond, le nouveau lien est ajouté. De la même manière, il est possible de vérifier les liens encore fonctionnels ou d'ajouter des liens à certains peptides n'en possédant pas. De cette manière nous avons remis à jour plusieurs dizaines de liens.

Mise à jour de la taxonomie La taxonomie pose plus de problèmes car elle évolue au cours du temps. La taxonomie du vivant et, en particulier, celle des bactéries a évolué au fur et à mesure des découvertes et des nouvelles méthodes de classification. Il n'y a aucun doute sur le fait qu'elle continuera à évoluer rapidement. Au sein de Norine, ce n'est pas une ressource critique mais il est bon de connaître les lignées d'espèces productrices de NRP pour pouvoir faire des rapprochements entre NRP/NRPS. Ces données de taxonomie peuvent être récupérées depuis le site ou l'API du NCBI. Le NCBI étant un organisme de référence, beaucoup de contributeurs maintiennent les classifications à jour. En effectuant régulièrement une recherche sur le NCBI de la taxonomie des espèces présentes dans Norine, il est possible de maintenir les données suffisamment à jour.

Les données de taxonomie de Norine sont, pour le moment, maintenues comme des chaînes de caractères au sein de la table des organismes. Cette structure est très redondante et ne permet pas de mises à jour efficaces. Nous avons profité de la création du script de mise à jour pour revoir complètement notre façon de stocker les taxons. La chaîne de caractères du champ organisme est remplacée par un identifiant dans la nouvelle table des taxons de Norine. Cette table est organisée comme un arbre pour permettre une mise à jour taxon par taxon sans avoir besoin de modifier des dizaines d'entrées. La construction de toute la chaîne taxonomique d'une espèce prend désormais beaucoup plus de temps (beaucoup plus de requêtes) mais ce problème est facilement contournable par la création de vues SQL. Par ce script, c'est l'ensemble des données de Norine qui ont été remises à jour.

Détection des changements de nomenclature Les deux scripts de mise à jour que nous venons de décrire ne peuvent pas fonctionner dans 100% des cas. Depuis leur publication (parfois remontant aux années 1980), certaines molécules et certains organismes ont changé de nom. Heureusement, à la fois le NCBI et PubChem gardent en mémoire une liste conséquente de synonymes permettant ainsi de les retrouver. Cependant, nos scripts ne trouvent pas toujours de synonymes, ce qui nous empêche d'effectuer les mises à jour. Dans ce cas, nous levons des alertes qui devront être analysées à la main (probablement durant des "Norine days").

Ajout de publications En analysant les annotations issues de s2m, nous nous

sommes parfois aperçu de l'existence de nouveaux articles à propos des molécules étudiées. Certains articles nous permettaient alors de mettre à jour l'annotation présente dans Norine, de créer des entrées pour des variants nouvellement connus et parfois d'invalider la production non ribosomique du peptide (dans ce cas le peptide n'a plus sa place dans Norine). Ce dernier cas est problématique car, pour le moment, il n'a pas été prévu dans Norine de retirer les informations d'une quelconque manière. Il est toujours possible de retirer les informations directement en SQL mais cette solution ne permet pas aux utilisateurs de comprendre la disparition de la molécule. Après concertation sur la nomenclature et les procédures d'évolution des données, nous nous sommes mis d'accord pour ajouter un statut "deprecated" au sein de la base. Pour le moment il existe les statuts "curated" et "putative" associés à chaque peptide. L'ajout de ce nouveau statut permettra de regrouper tous les peptides qui ont un jour été considérés NRP mais qui, depuis, ont été prouvés produits par d'autres voies de synthèse. Encore une fois, cette mise à jour n'est pas automatisable et nécessite un traitement particulier. La procédure devrait un jour être ajoutée à myNorine.

3.2.2.2 Récupération massive de NRP

L'exploration des bases de données proches des NRP permet de se rendre compte que de nombreuses molécules NRP connues ne sont pas répertoriées au sein de Norine. Afin d'essayer de combler le retard d'annotation de ces peptides, nous avons décidé d'automatiser leur intégration dans Norine. À partir d'un script, nous analysons et ajoutons les NRP présents au sein des bases MiBIG (voir 1.2.4) et BIRD. BIRD [?] (the Biologically Interesting molecule Reference Dictionary) est une base de données regroupant de petits polymères antibiotiques et inhibiteurs. Les deux bases ont pour avantage de contenir des entrées permettant la différentiation des NRP du reste des molécules. Cela nous facilite la tâche en nous assurant du statut des molécules que nous ajoutons à Norine.

Une fois les molécules récupérées, nous utilisons s2m pour produire l'annotation monomérique nécessaire à l'entrée d'un NRP en base. Cependant, en ajoutant automatiquement de nombreuses et nouvelles molécules automatiquement, nous ne respectons plus la charte de qualité que s'étaient fixés les créateurs de Norine. C'est pourquoi chacune des données importées automatiquement porte un statut nouveau appelé "automated". Lorsqu'un utilisateur recherchera des peptides sur l'interface web, les annotation manuelles seront mises en avant par rapport à celles générées automatiquement. Les pages de ces nouveaux peptides seront également marquées avec la provenance des données et un lien y apparaîtra.

Par ce script, 471 nouvelles annotations NRP vont être ajoutées à la prochaine ver-

sion de Norine (+40% de peptides). Une troisième partie de script est en cours d'intégration afin d'indexer les données issues de ClustScan database. Ce travail a été effectué en collaboration forte avec Juraj Michalik lorsqu'il était en contrat d'ingénieur au sein de notre équipe.

3.2.2.3 Fouille en bases de données

La dernière méthode de mise à jour de Norine est la plus exploratoire. L'idée est de scanner d'immenses bases de données chimiques à la recherche de peptides dont la structure monomérique nous fait penser à un NRP. Ces structures seront générées par s2m à partir des SMILES présents dans ces bases. L'optimisation effectuée sur le temps de calcul de s2m permet de passer à l'échelle et de scanner des dizaines de milliers de molécules. Sans aucune modification de s2m, nous sommes capables de charger et d'analyser environ 250 molécules par minute. En désactivant la recherche *light* pour tous les peptides qui n'ont pas une couverture suffisante, il est possible d'annoter 3 fois plus de structures dans le même temps.

Pour effectuer un premier test, nous avons choisi d'extraire de PubChem toutes les molécules qui répondent à la requête "peptide". Ce premier filtre limite le nombre d'entrées que nous avons à évaluer lors d'un premier test. Au total c'est 1 626 461 composés qui ont été récupérés. Les annotations de ces peptides par s2m ont été faites en un peu plus de 36 heures sur une seule machine standard (3.2 Ghz de CPU mono-thread et 8 Go de RAM dédiés au programme).

Une fois les annotations obtenues, nous cherchons à séparer les NRP du reste des molécules en utilisant plusieurs critères de filtration. Obtenir un bon taux de couverture lors du passage de s2m sera notre premier critère. Pour commencer, soyons très exigeants et réclamons de n'obtenir que les peptides 100% couverts. Nous obtenons 12830 peptides.

En regardant rapidement, on s'aperçoit que de nombreuses molécules sont constituées de seulement un ou deux monomères. Ajoutons un filtre sur le nombre minimal de monomères par molécule. Dans les statistiques de Norine, la plupart des peptides possèdent au moins 7 monomères. Filtrons donc toutes les annotations de moins de 7 monomères. Nous obtenons désormais 5342 peptides.

Enfin, les NRP sont noyés au milieu de nombreuses protéines classiques. Rappelons que les protéines classiques sont constituées uniquement de 20 acides aminés différents. Leur voie de synthèse n'effectue que des assemblages linéaires. Excepté quelques modifications enzymatiques post-synthèse, les peptides classiques sont linéaires. En filtrant tous

les peptides qui ne contiennent que des acides aminés protéogéniques ainsi que ceux qui sont linéaires, nous devrions augmenter la densité de NRP dans notre sortie. Nous ajoutons comme filtre l'impossibilité d'être linéaire (nombre de liaisons supérieur ou égal au nombre de nœuds) et l'obligation de posséder au moins deux monomères ne faisant pas partie des 20 de base. Finalement nous obtenons 1097 peptides.

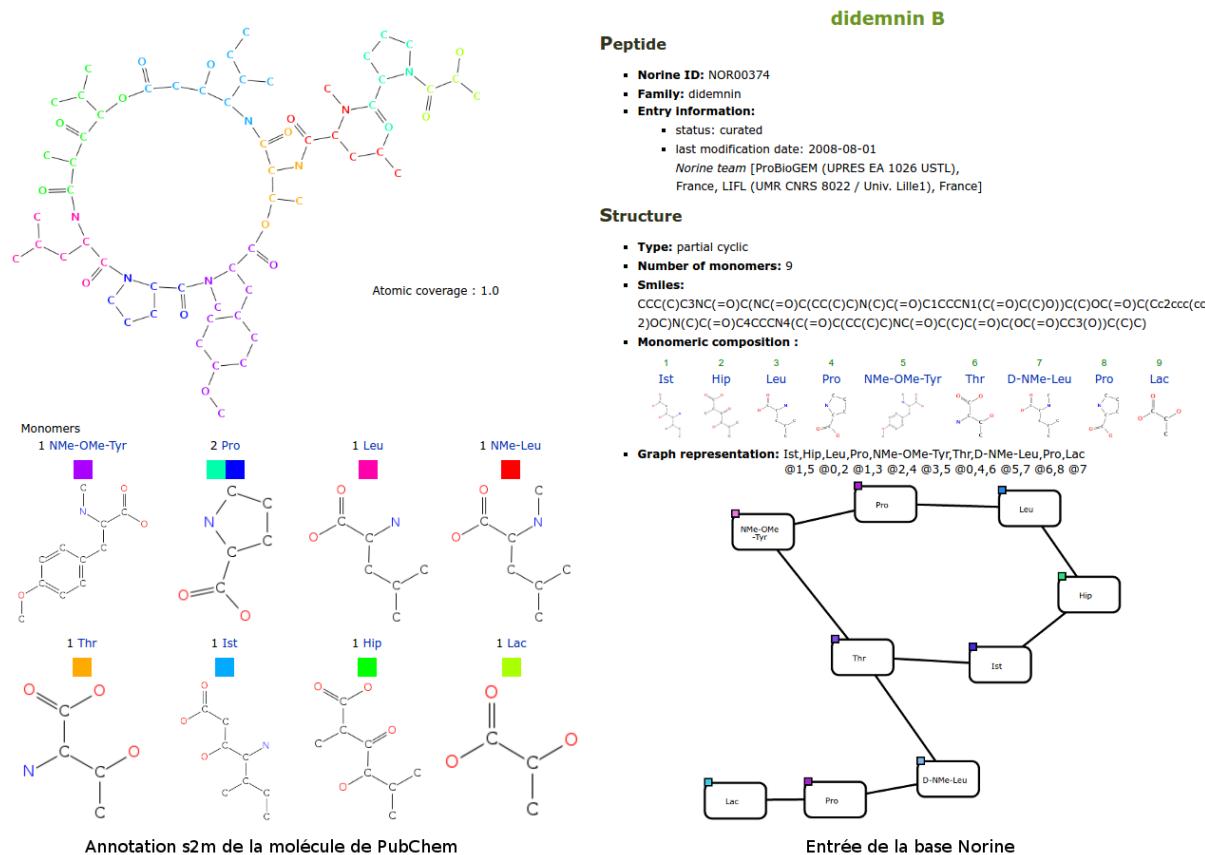


Figure 3.8: À gauche : une des sorties après processus de filtration. À droite : le NRP correspondant dans la base Norine.

Bien évidemment, ces 1097 structures ne sont pas toutes des peptides non ribosomiques. Les protéines classiques sont modifiées par de nombreuses enzymes qui permettent parfois la cyclisation et la transformation de monomères. Ces protéines peuvent être modifiées au point de les faire ressembler à des NRP. Nous ne pouvons donc pas être certains que les molécules issues du processus de filtration soient bien NRP. Cependant un bon indice de la qualité de la méthode est que nous retrouvons à l'aveugle des NRP de Norine. Nous avons par exemple retrouvé la “didemnin B” (figure 3.8). En effectuant une recherche par structure depuis l’interface de Norine, nous avons retrouvé ce NRP. Quelques exemples n’étant pas des preuves, il faudra revenir sur ces données afin de les valider manuellement. C'est à

nouveau un très long travail de bioanalyse qui viendra compléter les données de Norine.

3.3 Vers la biologie de synthèse

Depuis quelques années, les bactéries développent de plus en plus de résistances aux antibiotiques utilisés en médecine animale. Les NRP étant une source très riche d'antibiotiques, de nombreuses équipes essayent de créer, à partir de NRP, de nouvelles molécules contre lesquelles les micro-organismes ne présentent pas de résistance. C'est dans le but de contribuer à la création de ces nouvelles molécules que nous allons ici aborder la thématique de la biologie de synthèse. La biologie de synthèse est un domaine qui vise à modifier des organismes pour les pousser à effectuer des tâches que nous souhaitons. Dans notre cas, le but est de faire produire en masse par une bactérie, des NRP qui nous intéressent.

Cette section va décrire la suite du travail présenté précédemment. L'annotation NRP n'est pas une fin en soi mais une donnée précieuse pour comprendre l'assemblage moléculaire et de fait, les voies de synthèse. Le Graal de la biologie de synthèse appliquée aux NRP est de savoir comment produire nos propres assemblages monomériques en manipulant des gènes NRPS. Comme nous le verrons par la suite, cette question est très ambitieuse par rapport à l'avancée des connaissances actuelles. Notre première étape sera de mettre en relation les NRP avec les clusters de gènes capables de produire des peptides similaires. Cette information sera très utile pour aider au design de nouveaux NRP à partir de briques déjà existantes.

3.3.1 L'existant

Commençons par un tour d'horizon des travaux dans ce domaine. Nous nous appuierons sur des travaux de revue spécialisés pour les NRP [?]. Effectuons un survol des techniques en commençant par de petites modifications monomériques pour arriver finalement à la création totale d'un NRP.

3.3.1.1 Les modifications peptidiques

L'une des approches pour obtenir de nouvelles structures peptidiques est de faire varier des structures déjà produites. La structure peut être modifiée post-synthèse NRPS, par des enzymes présentes dans le cluster de gènes. Cela se produit naturellement, par exemple, lors d'incorporations de sucres dans un NRP (voir section 1.1.6.1). En ajoutant des gènes d'enzymes de modification NRP au cluster de gènes ou en modifiant l'activité d'enzymes déjà présentes, certaines équipes parviennent à l'ajout de nouveaux monomères ou la modification de certains monomères déjà inclus [?] (voir figure 3.9).

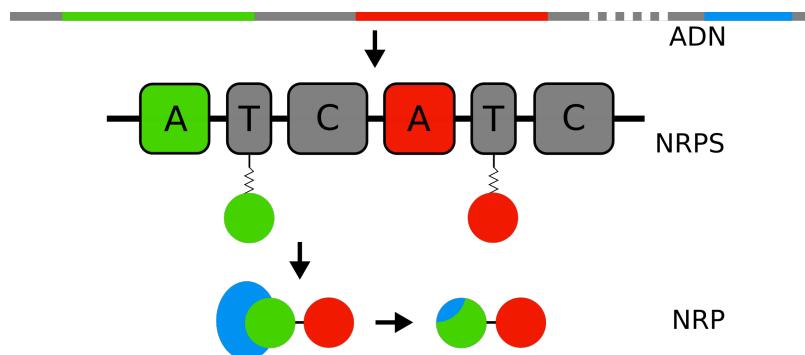


Figure 3.9: Modification post-synthèse du NRP par une action enzymatique. Les gènes NRPS contenant l'encodage des domaines A “vert” et “rouge” sont transcrits puis traduits depuis l'ADN afin de former les NRPS contenant les domaines de même couleur. La NRPS assemble ensuite le peptide. C'est enfin une enzyme ajoutée au cluster de gènes (ici en bleu), qui vient modifier l'un des monomères du NRP.

L'article [?] propose également d'aller plus loin en effectuant parfois cette modification avant l'assemblage NRPS. Les auteurs publient par exemple plusieurs modifications pré-synthèse de la Pacidamycin. Ce sont des enzymes modificatrices qui agissent sur un monomère cible avant sa capture (par exemple, dans l'article, Trp est dérivé en Cl-Trp ou d'autres variants). La structure du monomère n'étant pas fondamentalement changée, il est inclus par la synthétase sans changement préalable de celle-ci. Les contraintes structurelles des NRP n'autorisent pas toujours une modification post-synthèse de ce monomère. La modification pré-synthèse de monomères, lorsqu'elle n'affecte pas la capture du dit monomère, ouvre beaucoup de possibilités de modifications.

3.3.1.2 Les modifications ponctuelles de synthétases

La seconde possibilité de modification de NRP passe par la modification des synthétases. Plus exactement par la modification de la séquence génétique correspondant à un domaine A d'une synthétase. Les travaux [? ? ?] partent du constat (également fait par Stachelhaus [65]) que les domaines A sont spécifiques aux monomères qu'ils capturent. En modifiant certains acides aminés composant un domaine A, la spécificité de ce domaine peut changer (voir figure 3.10). Après quelques mutations bien effectuées, la poche d'accueil du monomère est suffisamment modifiée pour accueillir un autre monomère. Les modifications à effectuer sont découvertes via des techniques d'évolution dirigée. Les auteurs génèrent des mutations aléatoires (partiellement dirigées) sur les modules A puis ne gardent que ceux qui changent de fonction. Par ce procédé, ils arrivent à capturer des monomères

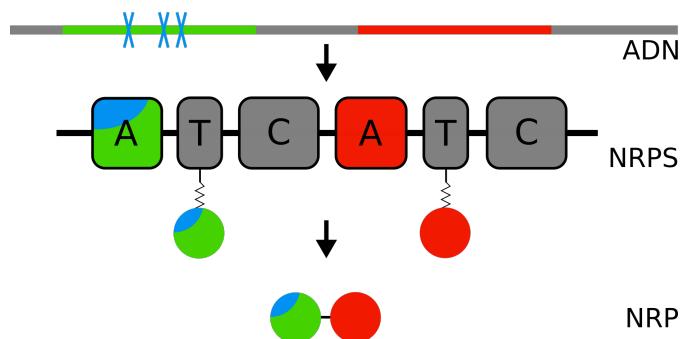


Figure 3.10: Des mutations ont été introduites au sein de l'ADN codant la NRPS, au niveau du domaine A vert. Le domaine A produit ne correspond plus à un domaine A vert mais à un variant proche. Le monomère capturé par ce domaine ne correspond plus à un monomère vert mais directement au variant proche correspondant au domaine modifié.

nouveaux, parfois non naturels [?].

3.3.1.3 Les modifications de domaines de synthétases

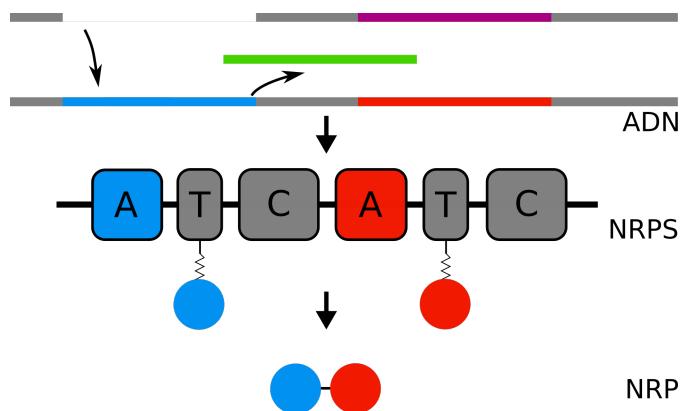


Figure 3.11: La séquence correspondant au domaine A incluant un monomère vert est remplacée par la séquence d'un domaine A correspondant à un monomère bleu. En conséquence, le monomère inclus par la synthétase est bleu.

Les modifications qui paraissent les plus évidentes sont celles qui font appel à des échanges de domaines (voir figure 3.11). Des exemples sont présentés au sein des articles [? ?]. Intuitivement, nous pourrions nous dire qu'il est facile d'échanger par exemple deux modules A de deux synthétases comme on échangerait deux pièces de LEGO® dans une construction. Malheureusement, ces modifications ne sont pas si directes car les modules échangés doivent respecter la structure 3D de la NRPS. Les structures 3D n'étant pas

souvent connues, il est très difficile de prédire un comportement de NRPS avant d'effectuer l'échange expérimentalement.

Bien que les contraintes structurelles soient des contraintes fortes, certaines équipes ont réussi des preuves de concept de cette technique. Par exemple, les auteurs de l'article [?] procèdent à l'échange d'une sous-partie de domaine A afin de modifier son site actif. Ils ont ainsi réussi à modifier le domaine A du premier module de la Gramicidine S et ainsi obtenir une production peptidique chimérique. Ce résultat est très encourageant pour la suite et devrait pouvoir mener à la production de variants de peptides existants.

3.3.1.4 La création *de novo* de peptides

Nous arrivons désormais au Graal de l'ingénierie NRP : créer un peptide de toutes pièces. Pour le moment, il n'existe aucune technique pour effectuer cette tâche. Comme nous l'avons vu précédemment, modifier un domaine A est déjà très difficile et nécessite de l'expertise accompagnée d'un peu de chance. Cependant, certains travaux cherchent déjà à concevoir des briques de base qui nous permettraient un jour de construire un peptide complet. Ainsi, les travaux menés par Wilcoxen *et al* [?] ont permis la création d'un peptide synthétique mimant le comportement d'un domaine C. Ils sont parvenus à lier deux monomères via leur peptide-domaine. Nous sommes encore loin d'un domaine C universel fonctionnel mais ces travaux permettront de mener aussi loin que possible cette quête du Graal.

Les techniques que nous avons présentées ici sont toutes très encourageantes pour l'avenir de la biologie de synthèse des NRP. Des preuves de concept expérimentales, complètes ou partielles (pour la création *de novo*), ont été réalisées dans chacune des directions que nous avons abordées. Cependant il reste plusieurs grands obstacles à surmonter pour parvenir à la création d'antibiotiques NRP synthétiques. Les différentes preuves de concept doivent être répliquées pour la fabrication de nombreux et divers NRP. C'est en multipliant les expériences que nous tirerons de l'information de ce qui fonctionne ou non. De plus, il faudra outrepasser les difficultés de quantité de production. Les divers NRP synthétiques sont créés avec des débits bien plus faibles que les NRP originaux. Il reste donc à comprendre comment dépasser les règles génétiques qui bloquent ces productions.

3.3.2 Aider les techniques de recombinaisons modulaires

Durant ma seconde année de thèse, j'ai pu séjourner un mois au Danemark dans l'équipe dirigée par Tilmann Weber afin de découvrir réciproquement nos travaux dans l'objectif

de renforcer notre collaboration. Son équipe a pour principale activité la découverte de nouveaux métabolites secondaires ainsi que la création d'outils bioinformatiques dédiés à ces découvertes. Cette équipe a créé, maintient et met à jour le logiciel antiSMASH (voir section 1.2.1.2). Pour rappel, antiSMASH est un logiciel qui analyse un génome pour générer des annotations de métabolites secondaires. C'est la découverte de NRP en particulier qui motiva mon voyage.

Mon objectif en me rendant dans cette équipe, était de préparer la création d'un outil qui effectue l'opération inverse de antiSMASH, c'est à dire remonter d'un peptide vers le cluster de gènes qui l'a produit. L'outil idéal (mais évidemment irréalisable) permettrait à partir d'un NRP de générer une séquence codante capable de produire ce NRP. C'est dans cette optique que j'ai créé un prototype pour la recommandation de clusters proches. Cette section sera dédiée à l'explication des prémisses de cet outil.

3.3.2.1 Mettre en relation NRP et gènes

Toutes les techniques expérimentales de modification de NRPS évoquées durant l'introduction sont très prometteuses mais encore en tout début de développement. Pour le moment, il n'existe pas de méthode de modification universelle mais uniquement des transformations au cas par cas. Pour contribuer aux avancées dans ce domaine, nous souhaitons créer un logiciel d'aide à la décision, permettant à partir d'un NRP souhaité, de trouver les clusters de gènes produisant une molécule la plus proche possible de ce NRP. Ainsi, nous espérons que parmi les molécules les plus proches, il existera un cluster NRPS modifiable pour synthétiser notre cible.

Pour concevoir notre logiciel de recommandation de NRP proches, nous pouvons nous inspirer de logiciels existants tels que Pep2Path [?]. Pep2Path est un logiciel qui permet de remonter les voies de synthèse pour aller découvrir les gènes NRPS responsables de la création d'un NRP. Leur idée est de faire de l'apprentissage sur les liens entre les pics observés dans des spectres de masse et les monomères prédits par NRPSPredictor2 depuis le cluster de gènes. Ainsi, lorsque le logiciel analyse un nouveau spectre, il se base sur les spectres déjà annotés pour pouvoir remonter au gène correspondant. De notre côté, dans une perspective de biologie de synthèse, nous souhaitons pouvoir utiliser une structure monomérique en entrée (et même une structure atomique) et non un spectre de masse mais nous pourrons tout-de-même utiliser des techniques proches.

Le logiciel devra prendre un NRP en entrée et fournir une liste de clusters de gènes en sortie, c'est-à-dire effectuer une recommandation (voir figure 3.12). Pour commencer simplement, nous étudierons uniquement la présence ou absence de domaines A dans la

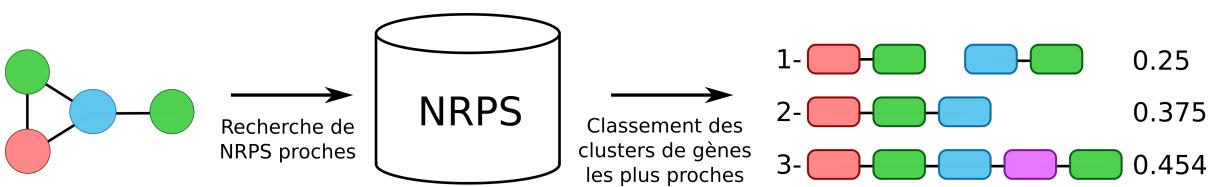


Figure 3.12: Processus de recommandation de clusters de gènes proches pour la création d'un NRP souhaité. Le logiciel prend en entrée un NRP que l'on souhaite synthétiser. Il doit chercher au sein d'une base de NRPS existantes, celles qui produisent le NRP le plus proche du NRP souhaité et classer les meilleurs résultats dans l'ordre de ressemblance. Ces clusters proches devraient pouvoir aider les biologistes à choisir des NRPS suffisamment proches du NRP souhaité pour pouvoir les dériver le plus facilement possible.

NRPS. À partir des différents domaines NRPS existants et des monomères présents dans le NRP d'intérêt nous établirons une distance. Cette distance permettra d'ordonner les clusters les uns par rapport aux autres. Prenons l'exemple d'un peptide d'intérêt contenant de nombreux monomères dont une unique leucine. Prenons également deux clusters de gènes identiques à l'exception d'un domaine A. Dans le premier cluster, ce domaine A inclus une leucine alors que le second non et aucun des autres domaines A ne recrute ce monomère. Alors, la distance que nous allons établir devra donner un meilleur score de proximité au premier cluster.

3.3.2.2 Recherche par fingerprint

L'approche que nous avons utilisé est une approche par fingerprint. En informatique, les fingerprints sont des structures représentant des données complexes par une projection de ces données à certaines de leurs caractéristiques essentielles. Dans notre cas, un fingerprint est un vecteur représentant la composition d'un NRP ou d'une NRPS. Dans le cas des NRP c'est la composition en monomères qui nous intéresse et dans le cas des NRPS la composition en domaines A. La composition peut être, par exemple, un simple comptage de chacun des monomères, un comptage de tous les dimères, de tous les trimères *etc.*

Dans notre cas, le vecteur sera composé des comptages de monomères seuls et des comptages de dimères (voir schéma 3.13). Pour le NRP, il est très simple d'obtenir ce vecteur. Chaque monomère est compté indépendamment puis chaque lien est compté comme un dimère comportant les deux monomères liés. Un dimère A-B est équivalent à un dimère B-A. Pour les NRPS, nous ne connaissons pas tous les voisinages. La connaissance actuelle du fonctionnement des enzymes NRPS ne nous permet pas tout le temps de prédire les

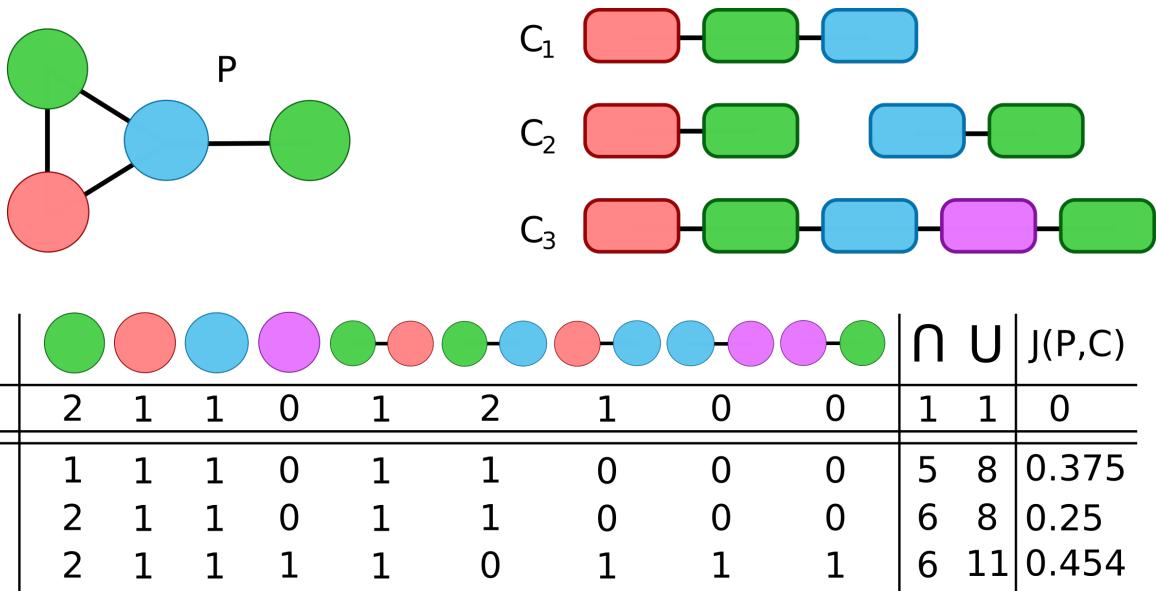


Figure 3.13: Exemple de calcul de distance entre un peptide P et trois clusters de gènes C_1 , C_2 et C_3 . Les clusters de gènes sont représentés par une suite de modules colorés en fonction de la couleur du monomère qui sera inclus. Dans le tableau sont présents les différents fingerprints pour le monomère et les clusters accompagnés du calcul de la distance de Jaccard.

emplacements exacts de cyclisation. Les règles générales d’assemblage de différents sous-peptides créés par différents gènes NRPS, ne sont pas facilement identifiables. La première partie des comptages est facile à obtenir. En analysant les domaines A, il est possible de déterminer leur spécificité (via par exemple NRPSPredictor) et ainsi d’augmenter la valeur de la case correspondant au monomère dans le vecteur. Pour ce qui est des dimères, il faut analyser les gènes NRPS. Deux domaines A dans deux gènes différents ne doivent pas générer de dimères. Nous ajoutons uniquement des dimères pour deux modules A successifs trouvés dans un même gène.

Une fois les vecteurs formés, il faut pouvoir comparer le fingerprint NRP aux fingerprints des clusters. Pour cela, nous avons choisi de comparer les fingerprints NRPS un à un avec le NRP et de calculer les distances les séparant. Cette distance permettra de classer les clusters par proximité et de ainsi proposer les plus proches du NRP. Pour générer cette distance il est nécessaire de prendre en compte à la fois les présences communes entre deux fingerprints mais également les divergences. Pour répondre à ce besoin, nous avons choisi d’utiliser une distance de Jaccard [?]. La distance de Jaccard compare la présence de critères communs entre les deux vecteurs par rapport au nombre de critères totaux. Dans notre cas nous comparerons les monomères et dimères communs à ceux divergents.

$$J(A, B) = 1 - \frac{A \cap B}{A \cup B} \quad (3.1)$$

Sur la figure 3.13, on peut voir l'application de cette formule à trois clusters de gènes. Prenons l'exemple du cluster C_1 . Ce cluster est composé de 3 domaines A chaînés : un rouge puis un vert et enfin un bleu. Le vecteur fingerprint représentant ce cluster est composé de 5 valeurs non nulles. Ces valeurs correspondent aux comptages de chacun des domaines A et des différents dimères de ces domaines. Ce vecteur diverge de celui correspondant au peptide par 1 vert et deux dimères manquants. Au total, son indice de Jaccard vaut $5/8$, ce qui donne une distance de Jaccard de $1 - 5/8 = 0.375$.

3.3.2.3 Tester le modèle

Afin de tester le modèle, nous avons choisi de prendre des peptides réels présents dans Norine. À partir de ces peptides, nous espérons retrouver leur vrai cluster de gènes au sein d'un ensemble de clusters. Pour cela, il nous est nécessaire de constituer une base de données test. Malheureusement, la constitution d'une telle base est très difficile. Il existe en effet très peu de sources de données mettant en relation des NRP constatés dans l'environnement avec des NRPS réelles. Nous devons constituer cette base par nous même.

La première piste que nous avions envisagée était de rassembler des données NRPS sûres (annotées manuellement) pour ne pas nous préoccuper de la qualité des données. Nous avons donc utilisé les données des couples gènes/NRP de MiBIG. L'intérêt des données issues de MiBiG sont qu'elles ont été entrées manuellement, c'est-à-dire que le contributeur avait suffisamment confiance en ses données pour les partager avec la communauté. Cependant, il est possible que le contributeur ait entré directement les résultats d'une prédiction sans ajouter d'annotations car celles-ci suffisaient pour ses travaux. Il nous faut donc vérifier la pertinence des structures NRP présentes en base. Pour cela nous avons effectué une validation croisée des peptides avec Norine. Malheureusement, sur les quelques centaines de couples NRP/NRP décrits dans MiBiG, seuls 60 NRP sont également présents dans Norine. Pour augmenter leur nombre, il sera nécessaire de valider d'autres structures manuellement. De plus, les données qui ont été soumises sont structurées différemment selon les contributeurs. MiBiG laisse une grande liberté dans les champs possibles à l'annotation, ce qui fait que plusieurs utilisateurs entrent des informations semblables sous différents noms. Cette diversité de structuration dans les fichiers d'annotation empêche la gestion automatique d'un grand nombre d'entrées. Suite au retrait des annotations impossibles à traiter dans un script (à moins de leur consacrer une exception), environ la moitié

des liens potentiels entre Norine et MiBIG ne sont plus présents. Ces deux contraintes cumulées nous ont poussé à changer de stratégie et ainsi changer notre jeu de données pour des données issues de prédictions.

A l'aide de Kai Blin (chercheur de l'équipe de Tilmann Weber), nous avons utilisé antiSMASH pour générer des annotations de NRPS sur l'ensemble des génomes de bactéries présentes sur le NCBI. Ce second jeu de données est beaucoup plus abondant que le premier mais pose également un problème de qualité. Il n'existe par exemple aucun lien entre les NRP de Norine et les génomes qui ont été analysés. Il faut faire manuellement les rapprochements pour connaître les binômes NRP/NRPS. De plus, les annotations de antiSMASH sont des prédictions et n'ont donc été vérifiées par personne. Par exemple, de nombreux petits clusters de gènes ne contiennent qu'un ou deux domaines A et il est difficile voire impossible de savoir s'ils sont artefactuels ou mal annotés.

Le programme a été lancé sur les jeux de données précédemment présentés dans cette section. Les problèmes évoqués pour ces jeux de données ne nous permettent pas pour le moment de dégager une quelconque conclusion sur la qualité de prédiction du modèle.

3.3.2.4 Perspectives

Ce travail est un travail en cours et les résultats présentés ici sont encore très préliminaires. Pour le moment, le modèle que nous avons développé ne prend en compte que les domaines A. Nous savons qu'une partie de l'information est portée par les autres domaines et c'est un critère qu'il nous faudra intégrer pour mieux rendre compte de la réalité. Par exemple, s2m permet de connaître le type de liaison entre deux monomères, ce qui n'est pas forcément le cas depuis une annotation manuelle à disposition en ligne. Cette connaissance pourrait être exploitée afin de rechercher les parties de NRPS effectuant ces liaisons. Nous supposons également que la distance de Jaccard n'est pas forcément très adaptée au problème et il nous faudra tenter l'utilisation d'alternatives issues des systèmes de recommandation classiques. Pour pouvoir continuer ce travail, il nous sera également nécessaire de constituer une base d'apprentissage de référence bien annotée. En effet, sans ces tests nous ne pourrons jamais déterminer avec précision la qualité des algorithmes développés. Nous sommes actuellement en train de réfléchir à des moyens de constituer ce jeu de données qui pourrait être d'une grande valeur pour de nombreuses tâches. La solution qui commence à émerger s'appuie sur plusieurs bases de données existantes. En utilisant à la fois Norine pour les NRP, MiBiG pour les annotations NRPS et le NCBI pour les séquences ADN, nous pouvons créer tout un système d'interconnexion donnant rapidement accès à toutes les informations. De notre côté nous commençons un long travail de recensement de NRPS correspondants aux

peptides de Norine. En soumettant sur MIBiG les annotations NRPS trouvées depuis et en concrétisant le lien depuis les pages des peptides, il nous est possible de démarrer ce jeu de données.

Conclusions et perspectives

Comme nous l'avons vu dès le premier chapitre, avant ce travail, il n'existe aucun moyen à la fois rapide et précis pour obtenir des annotations biologiques de peptides non ribosomiques. Les annotations étaient soit rapides mais issues de prédictions, soit précises car issues de données expérimentales (générées par exemple par spectrométrie), mais annotées à la main (section 1.2). Notre approche propose de venir compléter le processus de découverte de structures par spectrométrie en proposant smiles2monomers comme logiciel de création automatique des annotations biologiques (section 2.2). s2m a été développé dans l'optique d'être rapide, le plus précis possible dans l'annotation et non spécifique aux NRP. Le logiciel effectue une annotation d'un polymère à partir d'une liste de monomères candidats.

Pour construire s2m, nous avons découpé le problème d'annotation en deux phases d'analyse informatique. Dans une première étape, nous recherchons individuellement au sein du peptide chacun des monomères définis comme candidats par l'utilisateur (section 2.2.3.1). Les monomères ne sont pas recherchés tels quels mais dérivés en famille de résidus (section 2.4.2) pour pouvoir utiliser des techniques exactes de recherche de sous-graphe (section 2.4.1). Nous avons ensuite créé une structure d'indexation de ces résidus pour minimiser le temps d'exécution lors de la recherche (section 2.4.1.3). Cette optimisation nous permet d'obtenir jusqu'à 35% de gain en temps (section 2.5.3.1) lors des phases de recherche sans erreur, par rapport à l'état de l'art. Certains résidus n'étant pas trouvables par une recherche stricte, nous avons également proposé une recherche *light* beaucoup plus lente (section 2.4.4). Nous utilisons ce type de recherche uniquement en cas d'impossibilité de trouver une annotation complète à la suite d'une exécution stricte.

Dans une seconde étape, nous effectuons un pavage non chevauchant des résidus qui ont été trouvés lors de la première phase (section 2.4.3). Nous avons proposé une méthode exacte de pavage utilisant des résolutions par programmation linéaire (section 2.4.3.3) ou par une heuristique gloutonne localement raffinée (section 2.4.3.4). Comme nous nous y attendions, le pavage exact est bien plus lent que le pavage heuristique, mais nous avons

également prouvé que, dans le contexte de son utilisation, il pouvait donner de moins bons résultats que le pavage heuristique.

Globalement, s2m est un logiciel rapide et efficace (section 2.40), que nous avons utilisé comme contrôle qualité, ainsi que pour l'analyse de grands jeux de données. Avec un temps d'exécution moyen de 18 ms par annotation sur les jeux de données test (Norine et CCD), le logiciel est extrêmement rapide et peut analyser des centaines de milliers de molécules si nécessaire (section 2.5.5). De plus, avec 253 sur 342 annotations retrouvées à 100%, une sensibilité de 0.9 sur les données issues de Norine, 318 annotations trouvées sur 378 pour CCD et une sensibilité 0.97 sur ces mêmes données, les résultats sont d'excelente qualité. Enfin, comme nous l'avons vu durant l'analyse de ces résultats, ces chiffres sont une minoration de la qualité du logiciel car ils ne prennent pas en compte les erreurs découvertes au sein des jeux de données.

Les découvertes de diverses erreurs au sein des bases de données test nous ont démontré l'efficacité du logiciel et ont permis, via les corrections, d'augmenter la qualité globale des données de Norine (section 3.2). De plus, en ajoutant s2m dans le pipeline d'ajout de nouvelles entrées dans Norine, nous nous sommes prémunis de futures erreurs. Enfin, grâce à s2m, nous avons pu analyser en moins d'une journée, des centaines de milliers de peptides issus de bases généralistes pour trouver de potentiels NRP candidats (sections 3.2.2.2 et 3.2.2.3). s2m a généré de très importantes quantités de données qui vont désormais mener à l'entrée de nouvelles entrées de qualité dans Norine.

Actuellement, ce travail ouvre deux perspectives à court terme. Tout d'abord, en fin de manuscrit (section 3.3), j'ai présenté le début de création d'un outil qui a pour vocation d'aider la recherche en biologie de synthèse des NRP. Pour le moment, cet outil ne peut être testé à cause de la faible quantité des données à notre disposition. De plus, il est peu probable que le simple modèle présenté nous permette de prédire tout type de NRPS correctement. Cependant intégrer d'autres informations que les domaines A dans le modèle permettrait sans doute des prédictions précises mais également vérifierait et approfondirait nos connaissances des structures, des domaines et des liaisons inter-domaines NRPS. De plus la constitution d'une base de données (ou d'un ensemble de bases) liant toutes les étapes de la construction d'un NRP serait une ressource précieuse en soit. Ces données amorceraient très certainement des campagnes de recherches de nouvelles NRPS par apprentissage.

La seconde perspective à court terme est la diffusion de s2m. Nous avons identifié que plusieurs bases de données comme CCD [52] ou le consortium HELM [?] souhaitant main-

tenir en permanence leurs polymères sous les formes à la fois atomiques et monomériques. Sur son site web, HELM souligne particulièrement qu'il est difficile de trouver les structures monomériques dans les publications et bases de données publiques. L'un des défis qu'ils référencent sur leur site est la transformation automatique des formats chimiques vers les formats monomériques. s2m est parfaitement capable de relever ce défi puisqu'il a été conçu pour pouvoir fonctionner avec n'importe quel type de polymères.

À plus long terme, ce travail pourra être continué pour accroître à la fois les connaissances théoriques sur les NRP et NRPS tout en développant l'interaction avec des pharmacologues pour les aspects traitements antibactériens et antifongiques. La résistance aux antibiotiques est en train de devenir un problème majeur de santé publique et j'espère que ce travail pourra, au moins en partie, aider à la découverte de nouvelles substances d'intérêt pour la santé.

Bibliography

- [1] MIBiG: Minimum information about a biosynthetic gene cluster. URL <http://mibig.secondarymetabolites.org/>.
- [2] Problème du sac à dos. URL https://fr.wikipedia.org/w/index.php?title=Probl%C3%A8me_du_sac_%C3%A0_dos&oldid=123266424#Algorithme_glouton.
Page Version ID: 123266424.
- [3] Predictive toxicology.
- [4] Ammar Abdo, Ségolène Caboche, Valérie Leclère, Philippe Jacques, and Maude Pupin. A new fingerprint to predict nonribosomal peptides activity. 26(10):1187–1194, . ISSN 0920-654X, 1573-4951. doi: 10.1007/s10822-012-9608-4. URL <http://link.springer.com/article/10.1007/s10822-012-9608-4>.
- [5] Ammar Abdo, Valérie Leclère, Philippe Jacques, Naomie Salim, and Maude Pupin. Prediction of new bioactive molecules using a bayesian belief network. 54(1):30–36, . ISSN 1549-9596. doi: 10.1021/ci4004909. URL <http://dx.doi.org/10.1021/ci4004909>.
- [6] E. Akkoyunlu. The enumeration of maximal cliques of large graphs. 2(1):1–6. ISSN 0097-5397. doi: 10.1137/0202001. URL <http://pubs.siam.org/doi/abs/10.1137/0202001>.
- [7] Mikael R. Andersen, Jakob B. Nielsen, Andreas Klitgaard, Lene M. Petersen, Mia Zachariassen, Tilde J. Hansen, Lene H. Blicher, Charlotte H. Gotfredsen, Thomas O. Larsen, Kristian F. Nielsen, and Uffe H. Mortensen. Accurate prediction of secondary metabolite gene clusters in filamentous fungi. 110(1):E99–107. ISSN 1091-6490. doi: 10.1073/pnas.1205532110.
- [8] Brian O. Bachmann and Jacques Ravel. Chapter 8 methods for in silico prediction of microbial polyketide and nonribosomal peptide biosynthetic pathways from DNA

- sequence data. volume 458 of *Complex Enzymes in Microbial Natural Product Biosynthesis, Part A: Overview Articles and Peptides*, pages 181–217. Academic Press. URL <http://www.sciencedirect.com/science/article/pii/S0076687909048083>.
- [9] Carl J. Balibar, Frédéric H. Vaillancourt, and Christopher T. Walsh. Generation of d amino acid residues in assembly of arthrobactin by dual condensation/epimerization domains. 12(11):1189–1200. ISSN 1074-5521. doi: 10.1016/j.chembiol.2005.08.010. URL <http://www.sciencedirect.com/science/article/pii/S1074552105002668>.
- [10] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. 28(1):235–242. ISSN 0305-1048.
- [11] Sérgolène Caboche, Maude Pupin, Valérie Leclère, Arnaud Fontaine, Philippe Jacques, and Gregory Kucherov. NORINE: a database of nonribosomal peptides. 36: D326–D331. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gkm792. URL http://nar.oxfordjournals.org/content/36/suppl_1/D326.
- [12] Mark J. Calcott and David F. Ackerley. Portability of the thiolation domain in recombinant pyoverdine non-ribosomal peptide synthetases. 15:162. ISSN 1471-2180. doi: 10.1186/s12866-015-0496-3. URL <http://dx.doi.org/10.1186/s12866-015-0496-3>.
- [13] Maw-Shang Chang, Li-Hsuan Chen, and Ling-Ju Hung. Moderately exponential time algorithms for the maximum induced matching problem. pages 1–18. ISSN 1862-4472, 1862-4480. doi: 10.1007/s11590-014-0813-z. URL <http://link.springer.com/article/10.1007/s11590-014-0813-z>.
- [14] K. R. Conway and C. N. Boddy. ClusterMine360: a database of microbial PKS/NRPS biosynthesis. 41:D402–D407. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gks993. URL <http://nar.oxfordjournals.org/lookup/doi/10.1093/nar/gks993>.
- [15] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. 26(10):1367–1372. ISSN 0162-8828. doi: 10.1109/TPAMI.2004.75. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1323804>.
- [16] Pablo Cruz-Morales, Christian E. Martínez-Guerrero, Marco A. Morales-Escalante, Luis A. Yáñez-Guerra, Johannes F. Kopp, Jörg Feldmann, Hilda E. Ramos-Aboites, and Francisco Barona-Gómez. Recapitulation of the evolution of biosynthetic gene clusters reveals hidden chemical diversity on bacterial genomes. page 020503. doi: 10.1101/020503. URL <http://biorxiv.org/content/early/2015/07/06/020503>.

- [17] Marzio De Biasi. complexity theory - NP-hardness of covering with rectangular pieces (google hash code 2015 test round) - computer science stack exchange. URL <http://cs.stackexchange.com/questions/48775/np-hardness-of-covering-with-rectangular-pieces-google-hash-code-2015-test-round-49074>.
- [18] Janko Diminic, Jurica Zucko, Ida Trninic Ruzic, Ranko Gacesa, Daslav Hranueli, Paul F. Long, John Cullum, and Antonio Starcevic. Databases of the thiotemplate modular systems (CSDB) and their in silico recombinants (r-CSDB). 40(6):653–659. ISSN 1476-5535. doi: 10.1007/s10295-013-1252-z.
- [19] Frederic Dorn. Planar subgraph isomorphism revisited. URL <http://arxiv.org/abs/0909.4692>.
- [20] Hans-Christian Ehrlich and Matthias Rarey. Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review. 1(1):68–79. ISSN 1759-0884. doi: 10.1002/wcms.5. URL <http://onlinelibrary.wiley.com/doi/10.1002/wcms.5/abstract>.
- [21] David Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’95, pages 632–640. Society for Industrial and Applied Mathematics. ISBN 0-89871-349-8. URL <http://dl.acm.org/citation.cfm?id=313651.313830>.
- [22] Robert Finking and Mohamed A. Marahiel. Biosynthesis of nonribosomal peptides. 58(1):453–488. doi: 10.1146/annurev.micro.58.030603.123615. URL <http://dx.doi.org/10.1146/annurev.micro.58.030603.123615>.
- [23] Areski Flissi, Yoann Dufresne, Juraj Michalik, Laurie Tonon, Stéphane Janot, Laurent Noé, Philippe Jacques, Valérie Leclère, and Maude Pupin. Norine, the knowledgebase dedicated to non-ribosomal peptides, is now open to crowdsourcing. 44:D1113–1118. ISSN 1362-4962. doi: 10.1093/nar/gkv1143.
- [24] Xue Gao, Stuart W. Haynes, Brian D. Ames, Peng Wang, Linda P. Vien, Christopher T. Walsh, and Yi Tang. Cyclization of fungal nonribosomal peptides by a terminal condensation-like domain. 8(10):823–830. ISSN 1552-4450. doi: 10.1038/nchembio.1047. URL <http://www.nature.com/nchembio/journal/v8/n10/abs/nchembio.1047.html>.

- [25] Michael R Garey and David S Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman. ISBN 0-7167-1044-7 978-0-7167-1044-8 0-7167-1045-5 978-0-7167-1045-5.
- [26] Dario Ghersi and Mona Singh. molBLOCKS: decomposing small molecule sets and uncovering enriched fragments. 30(14):2081–2083. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btu173. URL <http://bioinformatics.oxfordjournals.org/content/30/14/2081>.
- [27] Andrea Grossi, Marco Locatelli, and Wayne Pullan. Simple ingredients leading to very efficient heuristics for the maximum clique problem. 14(6):587–612. ISSN 1381-1231, 1572-9397. doi: 10.1007/s10732-007-9055-x. URL <http://link.springer.com/article/10.1007/s10732-007-9055-x>.
- [28] MohammadTaghi Hajiaghayi and Naomi Nishimura. Subgraph isomorphism, log-bounded fragmentation, and graphs of (locally) bounded treewidth. 73(5):755–768. ISSN 0022-0000. doi: 10.1016/j.jcss.2007.01.003. URL <http://www.sciencedirect.com/science/article/pii/S0022000007000049>.
- [29] Saifuddin Kaijar and S. Durga Bhavani. Developing heuristic for subgraph isomorphism problem. In Manish Parashar, Dinesh Kaushik, Omer F. Rana, Ravi Samtaney, Yuanyuan Yang, and Albert Zomaya, editors, *Contemporary Computing*, number 306 in Communications in Computer and Information Science, pages 40–52. Springer Berlin Heidelberg. ISBN 978-3-642-32128-3 978-3-642-32129-0. URL http://link.springer.com/chapter/10.1007/978-3-642-32129-0_10. DOI: 10.1007/978-3-642-32129-0_10.
- [30] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US. ISBN 978-1-4684-2003-6 978-1-4684-2001-2. URL http://link.springer.com/chapter/10.1007/978-1-4684-2001-2_9. DOI: 10.1007/978-1-4684-2001-2_9.
- [31] Takeshi Kawabata. Build-up algorithm for atomic correspondence between chemical structures. 51(8):1775–1787. ISSN 1549-9596. doi: 10.1021/ci2001023. URL <http://dx.doi.org/10.1021/ci2001023>.
- [32] Nora Khaldi, Fayaz T. Seifuddin, Geoff Turner, Daniel Haft, William C. Nierman, Kenneth H. Wolfe, and Natalie D. Fedorova. SMURF: Genomic mapping of fungal sec-

- ondary metabolite clusters. 47(9):736–741. ISSN 1096-0937. doi: 10.1016/j.fgb.2010.06.003.
- [33] Rahul M. Kohli, Junichi Takagi, and Christopher T. Walsh. The thioesterase domain from a nonribosomal peptide synthetase as a cyclization catalyst for integrin binding peptides. 99(3):1247–1252. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.251668398. URL <http://www.pnas.org/content/99/3/1247>.
- [34] Andrew R. Leach. *Molecular modelling: principles and applications*. Prentice Hall, 2nd ed edition. ISBN 978-0-582-38210-7.
- [35] Gerald M Maggiora and Veerabahu Shanmugasundaram. Molecular similarity measures. 672:39–100. ISSN 1940-6029. doi: 10.1007/978-1-60761-839-3_2.
- [36] G. Manić, Laura Bahiense, and Cid de Souza. A branch&cut algorithm for the maximum common edge subgraph problem. 35:47–52. ISSN 1571-0653. doi: 10.1016/j.endm.2009.11.009. URL <http://www.sciencedirect.com/science/article/pii/S1571065309001620>.
- [37] Mohamed A. Marahiel, Torsten Stachelhaus, and Henning D. Mootz. Modular peptide synthetases involved in nonribosomal peptide synthesis. 97(7):2651–2674. ISSN 0009-2665, 1520-6890. doi: 10.1021/cr960029e. URL <http://pubs.acs.org/doi/abs/10.1021/cr960029e>.
- [38] James J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. 12(1):23–34. ISSN 1097-024X. doi: 10.1002/spe.4380120103. URL <http://onlinelibrary.wiley.com/doi/10.1002/spe.4380120103/abstract>.
- [39] Marnix H. Medema, Kai Blin, Peter Cimermancic, Victor de Jager, Piotr Zakrzewski, Michael A. Fischbach, Tilmann Weber, Eriko Takano, and Rainer Breitling. anti-SMASH: rapid identification, annotation and analysis of secondary metabolite biosynthesis gene clusters in bacterial and fungal genome sequences. 39:W339–346, . ISSN 1362-4962. doi: 10.1093/nar/gkr466.
- [40] Marnix H. Medema, Renzo Kottmann, Pelin Yilmaz, Matthew Cummings, John B. Biggins, Kai Blin, Irene de Bruijn, Yit Heng Chooi, Jan Claesen, R. Cameron Coates, Pablo Cruz-Morales, Srikanth Duddela, Stephanie Düsterhus, Daniel J. Edwards, David P. Fewer, Neha Garg, Christoph Geiger, Juan Pablo Gomez-Escribano, Anja Greule, Michalis Hadjithomas, Anthony S. Haines, Eric J. N. Helfrich, Matthew L. Hillwig, Keishi Ishida, Adam C. Jones, Carla S. Jones, Katrin Jungmann, Carsten Kegler,

Hyun Uk Kim, Peter Kötter, Daniel Krug, Joleen Masschelein, Alexey V. Melnik, Simone M. Mantovani, Emily A. Monroe, Marcus Moore, Nathan Moss, Hans-Wilhelm Nützmann, Guohui Pan, Amrita Pati, Daniel Petras, F. Jerry Reen, Federico Rosconi, Zhe Rui, Zhenhua Tian, Nicholas J. Tobias, Yuta Tsunematsu, Philipp Wiemann, Elizabeth Wyckoff, Xiaohui Yan, Grace Yim, Fengan Yu, Yunchang Xie, Bertrand Aigle, Alexander K. Apel, Carl J. Balibar, Emily P. Balskus, Francisco Barona-Gómez, Andreas Bechthold, Helge B. Bode, Rainer Borriss, Sean F. Brady, Axel A. Brakhage, Patrick Caffrey, Yi-Qiang Cheng, Jon Clardy, Russell J. Cox, René De Mot, Stefano Donadio, Mohamed S. Donia, Wilfred A. van der Donk, Pieter C. Dorrestein, Sean Doyle, Arnold J. M. Driessen, Monika Ehling-Schulz, Karl-Dieter Entian, Michael A. Fischbach, Lena Gerwick, William H. Gerwick, Harald Gross, Bertolt Gust, Christian Hertweck, Monica Höfte, Susan E. Jensen, Jianhua Ju, Leonard Katz, Leonard Kaysser, Jonathan L. Klassen, Nancy P. Keller, Jan Kormanec, Oscar P. Kuipers, Tomohisa Kuzuyama, Nikos C. Kyriides, Hyung-Jin Kwon, Sylvie Lautru, Rob Lavigne, Chia Y. Lee, Bai Linquan, Xinyu Liu, Wen Liu, Andriy Luzhetsky, Taifo Mahmud, Yvonne Mast, Carmen Méndez, Mikko Metsä-Ketelä, Jason Micklefield, Douglas A. Mitchell, Bradley S. Moore, Leonilde M. Moreira, Rolf Müller, Brett A. Neilan, Markus Nett, Jens Nielsen, Fergal O'Gara, Hideaki Oikawa, Anne Osbourn, Marcia S. Osburne, Bohdan Ostash, Shelley M. Payne, Jean-Luc Pernodet, Miroslav Petricek, Jörn Piel, Olivier Ploux, Jos M. Raaijmakers, José A. Salas, Esther K. Schmitt, Barry Scott, Ryan F. Seipke, Ben Shen, David H. Sherman, Kaarina Sivonen, Michael J. Smanski, Margherita Sosio, Evi Stegmann, Roderich D. Süßmuth, Kapil Tahlan, Christopher M. Thomas, Yi Tang, Andrew W. Truman, Muriel Viaud, Jonathan D. Walton, Christopher T. Walsh, Tilman Weber, Gilles P. van Wezel, Barrie Wilkinson, Joanne M. Willey, Wolfgang Wohlleben, Gerard D. Wright, Nadine Ziemert, Changsheng Zhang, Sergey B. Zotchev, Rainer Breitling, Eriko Takano, and Frank Oliver Glöckner. Minimum information about a biosynthetic gene cluster. 11(9):625–631, . ISSN 1552-4450. doi: 10.1038/ncchembio.1890. URL <http://www.nature.com/ncchembio/journal/v11/n9/full/ncchembio.1890.html>.

- [41] Hosein Mohimani and Pavel A. Pevzner. Dereplication, sequencing and identification of peptidic natural products: from genome mining to peptidogenomics to spectral networks. 33(1):73–86. ISSN 1460-4752. doi: 10.1039/C5NP00050E. URL <http://pubs.rsc.org/en/content/articlelanding/2016/np/c5np00050e>.
- [42] Hosein Mohimani, Wei-Ting Liu, Roland D. Kersten, Bradley S. Moore, Pieter C. Dorrestein, and Pavel A. Pevzner. NRPquest: Coupling mass spectrometry and genome

- mining for nonribosomal peptide discovery. 77(8):1902–1909. ISSN 0163-3864, 1520-6025. doi: 10.1021/np500370c. URL <http://pubs.acs.org/doi/abs/10.1021/np500370c>.
- [43] Katta G. Murty. *Linear programming*. Wiley. ISBN 978-0-471-09725-9.
- [44] NCBI Resource Coordinators. Database resources of the national center for biotechnology information. 41:D8–D20. ISSN 1362-4962. doi: 10.1093/nar/gks1189.
- [45] Samba Ndojh Ndiaye and Christine Solnon. CP models for maximum common subgraph problems. In Jimmy Lee, editor, *Principles and Practice of Constraint Programming – CP 2011*, number 6876 in Lecture Notes in Computer Science, pages 637–644. Springer Berlin Heidelberg. ISBN 978-3-642-23785-0 978-3-642-23786-7. URL http://link.springer.com/chapter/10.1007/978-3-642-23786-7_48. DOI: 10.1007/978-3-642-23786-7_48.
- [46] Jiří Novák, Karel Lemr, Kevin A. Schug, and Vladimír Havlíček. CycloBranch: De novo sequencing of nonribosomal peptides from accurate product ion mass spectra. 26(10):1780–1786. ISSN 1044-0305, 1879-1123. doi: 10.1007/s13361-015-1211-1. URL <http://link.springer.com/article/10.1007/s13361-015-1211-1>.
- [47] Noel M O’Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open babel: An open chemical toolbox. 3(1):33. ISSN 1758-2946. doi: 10.1186/1758-2946-3-33. URL <http://jcheminf.springeropen.com/articles/10.1186/1758-2946-3-33>.
- [48] James B Orlin. Line-digraphs, arborescences, and theorems of tutte and knuth. 25(2):187–198. ISSN 0095-8956. doi: 10.1016/0095-8956(78)90038-2. URL <http://www.sciencedirect.com/science/article/pii/0095895678900382>.
- [49] George A. Patani and Edmond J. LaVoie. Bioisosterism: A rational approach in drug design. 96(8):3147–3176. ISSN 0009-2665. doi: 10.1021/cr950066q. URL <http://dx.doi.org/10.1021/cr950066q>.
- [50] M. Pelillo, K. Siddiqi, and S. W. Zucker. Matching hierarchical structures using association graphs. 21(11):1105–1120. ISSN 0162-8828. doi: 10.1109/34.809105.
- [51] S. W. Queener. Molecular biology of penicillin and cephalosporin biosynthesis. 34(6):943–948. ISSN 0066-4804.

- [52] Syed A. Rahman, Matthew Bashton, Gemma L. Holliday, Rainer Schrader, and Janet M. Thornton. Small molecule subgraph detector (SMSD) toolkit. 1(1):12. ISSN 1758-2946. doi: 10.1186/1758-2946-1-12. URL <http://www.jcheminf.com/content/1/1/12/abstract>.
- [53] Christian Rausch, Ilka Hoof, Tilmann Weber, Wolfgang Wohlleben, and Daniel H. Huson. Phylogenetic analysis of condensation domains in NRPS sheds light on their functional evolution. 7:78, . ISSN 1471-2148. doi: 10.1186/1471-2148-7-78. URL <http://dx.doi.org/10.1186/1471-2148-7-78>.
- [54] Christian Rausch, Tilmann Weber, Oliver Kohlbacher, Wolfgang Wohlleben, and Daniel H. Huson. Specificity prediction of adenylation domains in nonribosomal peptide synthetases (NRPS) using transductive support vector machines (TSVMs). 33(18):5799–5808, . ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gki885. URL <http://nar.oxfordjournals.org/content/33/18/5799>.
- [55] John W. Raymond and Peter Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. 16(7):521–533. ISSN 0920-654X, 1573-4951. doi: 10.1023/A:1021271615909. URL <http://link.springer.com/article/10.1023/A%3A1021271615909>.
- [56] Marc Röttig, Marnix H. Medema, Kai Blin, Tilmann Weber, Christian Rausch, and Oliver Kohlbacher. NRPSpredictor2—a web server for predicting NRPS adenylation domain specificity. 39:W362–W367. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gkr323. URL http://nar.oxfordjournals.org/content/39/suppl_2/W362.
- [57] Dirk Schwarzer, Robert Finking, and Mohamed A. Marahiel. Nonribosomal peptides: from genes to products. 20(3):275. ISSN 0265-0568, 1460-4752. doi: 10.1039/b111145k. URL <http://xlink.rsc.org/?DOI=b111145k>.
- [58] Georg Schönafinger. Amide bond formation in nonribosomal peptide synthesis: The formylation and condensation domains. URL <http://archiv.ub.uni-marburg.de/diss/z2008/0068/pdf/dgs.pdf>.
- [59] R. Shamir and D. Tsur. Faster subtree isomorphism. In , *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, 1997, pages 126–131. doi: 10.1109/ISTCS.1997.595164.
- [60] Ben Shen. Polyketide biosynthesis beyond the type I, II and III polyketide synthase

- paradigms. 7(2):285–295. ISSN 1367-5931. doi: 10.1016/S1367-5931(03)00020-6. URL <http://www.sciencedirect.com/science/article/pii/S1367593103000206>.
- [61] Michael A. Siani, David Weininger, and Jeffrey M. Blaney. CHUCKLES: A method for representing and searching peptide and peptoid sequences on both monomer and atomic levels. 34(3):588–593. ISSN 0095-2338. doi: 10.1021/ci00019a017. URL <http://dx.doi.org/10.1021/ci00019a017>.
- [62] Hyuk-Hwan Song, Hee-Seok Lee, Jin-Ho Jeong, Hee-Seung Park, and Chan Lee. Diversity in beauvericin and enniatins h, i, and MK1688 by fusarium oxysporum isolated from potato. 122(3):296–301. ISSN 0168-1605. doi: 10.1016/j.ijfoodmicro.2008.01.009.
- [63] T. Stachelhaus, A. Hüser, and M. A. Marahiel. Biochemical characterization of peptidyl carrier protein (PCP), the thiolation domain of multifunctional peptide synthetases. 3 (11):913–921, . ISSN 1074-5521.
- [64] T. Stachelhaus, H. D. Mootz, V. Bergendahl, and M. A. Marahiel. Peptide bond formation in nonribosomal peptide biosynthesis. catalytic role of the condensation domain. 273(35):22773–22781, . ISSN 0021-9258.
- [65] T. Stachelhaus, H. D. Mootz, and M. A. Marahiel. The specificity-conferring code of adenylation domains in nonribosomal peptide synthetases. 6(8):493–505, . ISSN 1074-5521. doi: 10.1016/S1074-5521(99)80082-9.
- [66] James Staunton and Kira J. Weissman. Polyketide biosynthesis: a millennium review. 18(4):380–416. ISSN 02650568, 14604752. doi: 10.1039/a909079g. URL <http://xlink.rsc.org/?DOI=a909079g>.
- [67] Christoph Steinbeck, Yongquan Han, Stefan Kuhn, Oliver Horlacher, Edgar Luttmann, and Egon Willighagen. The chemistry development kit (CDK): An open-source java library for chemo- and bioinformatics. 43(2):493–500. ISSN 0095-2338. doi: 10.1021/ci025584y. URL <http://pubs.acs.org/doi/abs/10.1021/ci025584y>.
- [68] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques. In Kyung-Yong Chwa and J. Ian J. Munro, editors, *Computing and Combinatorics*, number 3106 in Lecture Notes in Computer Science, pages 161–170. Springer Berlin Heidelberg. ISBN 978-3-540-22856-1 978-3-540-27798-9. URL http://link.springer.com/chapter/10.1007/978-3-540-27798-9_19. DOI: 10.1007/978-3-540-27798-9_19.

- [69] John W. Trauger, Rahul M. Kohli, Henning D. Mootz, Mohamed A. Marahiel, and Christopher T. Walsh. Peptide cyclization catalysed by the thioesterase domain of tyrocidine synthetase. 407(6801):215–218. ISSN 0028-0836. doi: 10.1038/35025116. URL <http://www.nature.com/nature/journal/v407/n6801/abs/407215a0.html>.
- [70] J. R. Ullmann. An algorithm for subgraph isomorphism. 23(1):31–42. ISSN 00045411. doi: 10.1145/321921.321925. URL <http://portal.acm.org/citation.cfm?doid=321921.321925>.
- [71] Myco Umemura, Hideaki Koike, and Masayuki Machida. Motif-independent de novo detection of secondary metabolite gene clusters-toward identification from filamentous fungi. 6:371. ISSN 1664-302X. doi: 10.3389/fmicb.2015.00371.
- [72] AndréM. A. van Wageningen, Peter N. Kirkpatrick, Dudley H. Williams, Barbara R. Harris, Jo K. Kershaw, Nicola J. Lennard, M. Jones, Steven J. M. Jones, and Patricia J. Solenberg. Sequencing and analysis of genes involved in the biosynthesis of a vancomycin group antibiotic. 5(3):155–162. ISSN 1074-5521. doi: 10.1016/S1074-5521(98)90060-6. URL <http://www.sciencedirect.com/science/article/pii/S1074552198900606>.
- [73] Yan Wang, Tyler W. H. Backman, Kevin Horan, and Thomas Girke. fmcsR: mismatch tolerant maximum common substructure searching in r. 29(21):2792–2794. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btt475. URL <http://bioinformatics.oxfordjournals.org/content/29/21/2792>.
- [74] T. Weber, C. Rausch, P. Lopez, I. Hoof, V. Gaykova, D. H. Huson, and W. Wohlleben. CLUSEAN: A computer-based framework for the automated analysis of bacterial secondary metabolite biosynthetic gene clusters. 140(1):13–17, . ISSN 0168-1656. doi: 10.1016/j.jbiotec.2009.01.007. URL <http://www.sciencedirect.com/science/article/pii/S0168165609000078>.
- [75] Tilmann Weber and Hyun Uk Kim. The secondary metabolite bioinformatics portal: Computational tools to facilitate synthetic biology of secondary metabolite production. 1(2):69–79. ISSN 2405-805X. doi: 10.1016/j.synbio.2015.12.002. URL <http://www.sciencedirect.com/science/article/pii/S2405805X15300156>.
- [76] Tilmann Weber, Kai Blin, Srikanth Duddela, Daniel Krug, Hyun Uk Kim, Robert Brucoleri, Sang Yup Lee, Michael A. Fischbach, Rolf Müller, Wolfgang Wohlleben, Rainer Breitling, Eriko Takano, and Marnix H. Medema. antiSMASH 3.0-a comprehensive

- resource for the genome mining of biosynthetic gene clusters. 43:W237–243, . ISSN 1362-4962. doi: 10.1093/nar/gkv437.
- [77] John D. Westbrook, Chenghua Shao, Zukang Feng, Marina Zhuravleva, Sameer Venkankar, and Jasmine Young. The chemical component dictionary: complete descriptions of constituent molecules in experimentally determined 3d macromolecules in the protein data bank. 31(8):1274–1278. ISSN 1367-4811. doi: 10.1093/bioinformatics/btu789.
- [78] Peter Willett. Similarity searching using 2d structural fingerprints. In Jürgen Bajorath, editor, *Chemoinformatics and Computational Chemical Biology*, number 672 in Methods in Molecular Biology, pages 133–158. Humana Press. ISBN 978-1-60761-838-6 978-1-60761-839-3. URL http://link.springer.com/protocol/10.1007/978-1-60761-839-3_5.
- [79] Thomas Wolf, Vladimir Shelest, Neetika Nath, and Ekaterina Shelest. CASSIS and SMIPS: promoter-based prediction of secondary metabolite gene clusters in eukaryotic genomes. 32(8):1138–1143. ISSN 1367-4811. doi: 10.1093/bioinformatics/btv713.
- [80] Laurence A. Wolsey. Mixed integer programming. In *Wiley Encyclopedia of Computer Science and Engineering*. John Wiley & Sons, Inc. ISBN 978-0-470-05011-8. URL <http://onlinelibrary.wiley.com/doi/10.1002/9780470050118.ecse244/abstract>.
- [81] Atsuko Yamaguchi, Kiyoko F. Aoki, and Hiroshi Mamitsuka. Finding the maximum common subgraph of a partial k-tree and a graph with a polynomially bounded number of spanning trees. 92(2):57–63. ISSN 0020-0190. doi: 10.1016/j.ipl.2004.06.019. URL <http://www.sciencedirect.com/science/article/pii/S0020019004002005>.
- [82] Xifeng Yan, Philip S. Yu, and Jiawei Han. Substructure similarity search in graph databases. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’05, pages 766–777. ACM. ISBN 1-59593-060-4. doi: 10.1145/1066157.1066244. URL <http://doi.acm.org/10.1145/1066157.1066244>.
- [83] Xihou Yin and T. Mark Zabriskie. The enduracidin biosynthetic gene cluster from streptomyces fungicidicus. 152(10):2969–2983. doi: 10.1099/mic.0.29043-0. URL <http://mic.microbiologyresearch.org/content/journal/micro/10.1099/mic.0.29043-0>.

- [84] Nadine Ziemert, Sheila Podell, Kevin Penn, Jonathan H. Badger, Eric Allen, and Paul R. Jensen. The natural product domain seeker NaPDoS: a phylogeny based bioinformatic tool to classify secondary metabolite gene diversity. 7(3):e34064. ISSN 1932-6203. doi: 10.1371/journal.pone.0034064.